2002

The fusion and integration of virtual sensors

Thomas F. Litant
College of William & Mary - Arts & Sciences

THE FUSION AND INTEGRATION OF VIRTUAL SENSORS

A Dissertation

Presented to

The Faculty of the Department of Computer Science

The College of William and Mary in Virginia

In Partial Fulfillment

Of the Requirements for the Degree of

Doctor of Philosophy

---

by

Thomas F. Litant

2002

APPROVAL SHEET

This dissertation is submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

Author

Approved, 9 January 2002

William L. Bynum

Robert E. Noonan

Stefan Feyock

Andreas Stathopoulos

Lawrence M. Leemis
Department of Mathematics

Table of Contents

Acknowledgements

The author gratefully acknowledges the patience, support, wisdom, and generosity of his graduate advisor: William Bynum. It is the great good fortune of succeeding classes at William and Mary that Dr. Bynum chose, after retirement, to remain as Emeritus Professor of Computer Science.

This dissertation is dedicated to the memories of Raquel Ethel Shafran Litant: Artist and Educator, and Irving Litant: Research Scientist. It is hoped that this dissertation contains a little of each.

This dissertation is also dedicated to the memory of the author's first Ph.D. advisor: D. Paul Snyder. A Philosopher of Science, Dr. Snyder was fond of saying: "The best we can hope for is to be wrong in interesting ways."

List of Tables

List of Figures

Abstract

There are numerous sensors from which to choose when designing a mobile robot: ultrasonic, infrared, radar, or laser range finders, video, collision detectors, or beacon based systems such as the Global Positioning System. In order to meet the need for reliability, accuracy, and fault tolerance, mobile robot designers often place multiple sensors on the same platform, or combine sensor data from multiple platforms. The combination of the data from multiple sensors to improve reliability, accuracy, and fault tolerance is termed Sensor Fusion.

The types of robotic sensors are as varied as the properties of the environment that need to be sensed. To reduce the complexity of system software, Roboticists have found it highly desirable to adopt a common interface between each type of sensor and the system responsible for fusing the information. The process of abstracting the essential properties of a sensor is called Sensor Virtualization.

Sensor virtualization to date has focused on abstracting the properties shared by sensors of the same type. The approach taken by T. Henderson is simply to expose to the fusion system only the data from the sensor, along with a textual label describing the sensor. We extend Henderson's work in the following manner. First, we encapsulate both the fusion algorithm and the interface layer in the virtual sensor. This allows us to build multi-tiered virtual sensor hierarchies. Secondly, we show how common fusion algorithms can be encapsulated in the virtual sensor, facilitating the integration and replacement of both physical and virtual sensors. Finally, we provide a physical proof of concept using monostatic sonars, vector sonars, and a laser range-finder.

THE FUSION AND INTEGRATION OF VIRTUAL SENSORS

Chapter 1

Overview

The fundamental research area in mobile robots is Navigation. Whether the navigation is structured, with defined beginning, end, and waypoints, or task oriented, such as roaming around while mapping obstructions, the mobile robot must solve three basic problems. First, it must determine a course, second it must determine its location globally, third, after moving, it must determine how far it has gone and in what direction. These problems are known as Route Planning, Local Localization, and Global Localization, respectively. Some of the earliest work involved Route Planning, and while it still generates the occasional scholarly research, is generally considered a thoroughly investigated field. Localization, however, remains as a fertile area for research. The route planner may instruct the mobile robot controller to rotate about its base and then move off a certain distance, but frequently the actual endpoint of this motion differs from the planned endpoint. There are a number of reasons for this difference: power fluctuations, friction, hardware failure, and so on. In order to reacquire its location, the robot must be designed to acquire off-board data, which is accomplished by sensors.

There are numerous sensors from which to choose designing a mobile robot: ultrasonic, infrared, radar, or laser range finders, video, collision detectors, or beacon based systems such as the Global Positioning System. In order to meet the need for reliability,

accuracy, and fault tolerance, mobile robot designers often place multiple sensors on the same platform. The combination of the data from multiple sensors to improve reliability, accuracy, and fault tolerance is termed Sensor Fusion.

Sensor Fusion dates back to at least the 1950s, when air defense systems designers developed the first algorithms to fuse radar data from multiple radar sites into a common picture of the air space. It is only within the last ten years that researchers in the field of mobile robots have seen sensor fusion to be a useful approach to robot navigation. At least part of the reason for this is that, until fairly recently, multiple sensor packages have been too expensive or too large to incorporate into a mobile platform. Economics still drives the sensor selection process to the extent that precision and reliability is roughly proportional to the cost of the sensor system. Thus another feature of sensor fusion is that it holds the promise of improving the performance of inexpensive sensors by improving the sensor data processing algorithms.

There are two ways of ordering sensor data before submitting them to the fusion process. First, we can look at the sensor data from multiple sensors at the same time (**concurrent fusion**). These sensors may be of the same type (**homogeneous**), or they may be of different types (**heterogeneous**). In either case, the objective is to come up with a single estimate that improves on the data provided by any single sensor.

The second approach is to time-order (**sequentially fuse**) the sensor data. Once again it is assumed that the set of sensor data will cluster about the correct values. Since

this form of fusion incorporates time, it is also used to estimate and to predict dynamic properties, such as the trajectory of a moving object.

Key to sensor fusion is the redundancy of the measurements. If sensor S1 reports that the target range is 15 meters, a report from sensor S2 that the target is blue does nothing to improve the accuracy of the range measurement unless, somehow, the color of the target can be transformed into a range measurement. If this can be done then S1 and the transformed data from S2 are commensurate, otherwise they are incommensurate and cannot be fused.

The types of robotic sensors are as varied as the properties of the environment that need to be sensed. To reduce the complexity of system software, roboticists have found it highly desirable to adopt a common interface between each type of sensor and the system. The process of abstracting the essential properties of a sensor is called sensor **virtualization**. The only authors to date to systematically explore sensor virtualization are Henderson and Shilcrat. Their main focus, however, is to create a system for automatically generating an interface description for physical sensors.[1] The approach taken by Henderson [45] is simply to expose to the **fusor** only the data from the sensor, along with a textual label describing the sensor. This approach fails, however, when we attempt to fuse this information. Many fusion algorithms require additional information about the

[1] This is called "logical sensor specification."

reliability or accuracy of the data. With this information, we can, accordingly, weigh unreliable data less heavily when fusing it with more reliable data.

Henderson's program is thus, incomplete, because it is performed in a vacuum: he does not adequately consider the requirements of the fusion algorithm that receives data from the virtual sensor.

The virtual sensor is more than Henderson's logical sensors in another respect. If the fusion algorithm or fusor can be "wrapped" so that it accepts input from a virtual sensor data vector and, in turn, outputs a data vector consistent with the virtual sensor interface specification, then the fusor itself can be treated as a virtual sensor. This modularization has interesting architectural implications; starting with layer n, the physical sensor layer, a tree of virtual sensors can be constructed, with each virtual sensor employing appropriate, and potentially different, fusion algorithms to the data received from subordinate layers. It has the potential to be very robust in the event of sensor failure or degradation in sensor accuracy. It can even be constructed to allow dynamic reconfiguration: adding and deleting sensors as the mission requires or sensor resource availability allows. (See Figure 1.)

Another benefit of virtualization is that, because the fusion algorithm is no longer inextricably linked to the physical sensor type, the designer is allowed wide latitude in selecting fusion algorithms. This selection can even be made dynamically to support changing mission requirements. This is essential because the preponderance of the evi-

dence points to the conclusion that there is no single fusion algorithm that is "best" under all circumstances.[2]

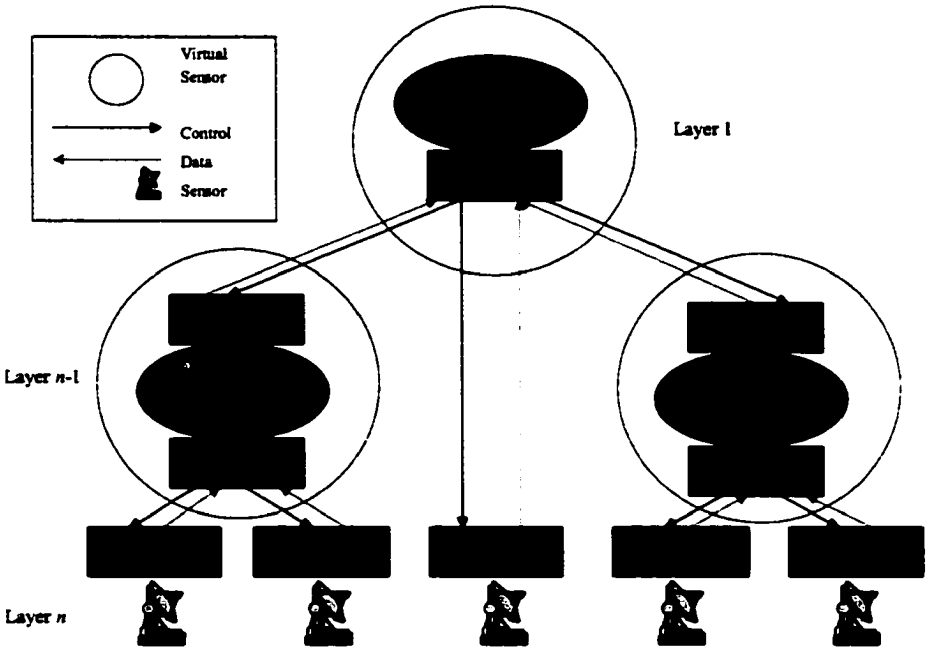

Figure 1: Virtual Sensor Architecture[3]

The objective of this dissertation is to formulate a virtual sensor interface (sensor virtualization), modifying common fusion algorithms where required to accept data from the virtual sensor and, in turn, to appear to the next layer as a virtual sensor (algorithm

---

[2] See Appendix F for a survey of studies comparing fusion algorithms.

[3] Because this depiction shows the virtual sensor architecture as hierarchical should not be taken to restrict it thus. A peer-to-peer architecture is also supported, where the virtualization layer of one virtual sensor feeds the input wrapper of another, and vice-versa.

wrapping). We will focus on pixel and signal level fusion (see Chapter 2), though we will describe how virtualization is easily extended to feature and symbol level fusion. As a side benefit, sensor virtualization brings to light limitations and misconceptions in some of the more common fusion methodologies; we provide several improvements to these in the process of algorithm wrapping. Finally, we demonstrate both the feasibility of sensor virtualization in a proof of concept while acknowledging its limitations under the current formulation. A novel approach to removing these limitations is described in a proposal for follow-on research (see Chapter 7).

Chapter 2

Taxonomy

To a certain extent some of the terminology used in this proposal is peculiar to sensor science and sensor fusion, and may not be familiar to those outside of these fields. Even in the case of terms common to these fields some redefinition has become necessary due to either a lack of a common consensus, or vagueness in definition.

First, sensor data can be categorized as **commensurate** or **incommensurate**. Data is commensurate when it measures the same properties of an object: range, location, size, weight, temperature, velocity, and so on, or if a function can be defined that maps the measurements from one property to another. For example, acceleration can be computed from velocity by calculating the rate of change, and thus the output from a sensor that measures velocity can be made commensurate with data from an accelerometer. Redundant data is, by definition, commensurate.

Sensor fusion is any algorithm that combines commensurate sensor data for the purpose of minimizing measurement error. This has often been confused with **object recognition** ("Automatic Target Recognition," "Correlation," etc.) where data is combined for the purpose of zeroing in on the correct object type from a set of possibilities.

One distinguishing characteristic of sensor fusion algorithms is that they require commensurate data. Object recognition, on the other hand, benefits little from commensurate, or redundant data. In either case the purpose of combining this sensor data is to increase some measure of confidence in the identity or properties of the object targeted beyond that which may be had with a single sensor.

In certain cases, sensor data concerning certain properties of the target can be manipulated or combined to produce information on another property. For example, range data from a sonar array can be used to detect features such as a wall, a corner, or a doorway. (e.g. [10]) This process of converting sensor data from one property to another will be called **sensor transformation**. Mathematically, this process may be a linear or a nonlinear function, or it may be a probabilistic function, such as conditional or joint probability distribution.

This dissertation will examine three methods for passing parameters from virtual sensor to virtual sensor. The first, **propagation** will refer to the passing of parameters from virtual sensor to virtual sensor, undergoing some transformation in the process. If no transformation takes place, then the parameters are merely **relayed**. The third method is where the parameters are not passed from the sending virtual sensor to the receiving virtual sensor, but are **generated** at the latter sensor.

Next, targets and sensors will be located in two or three space by Cartesian coordinates we will refer to as **position**, while the angular relation of the target or sensor in

relation to some frame of reference will be termed **pose**. **Sensor registration** is the process of aligning the field of view of multiple sensors, while **sensor mensuration** is the process determining the source coordinates for a particular piece of sensor **data. Coverage analysis** is the term used for algorithms used to determine what targets should be detectable give the position, field of view, and detection characteristics of the available sensors.

Finally, we describe four hierarchical levels of sensor fusion: **signal, pixel, feature,** and **symbol.** A **signal** is a waveform or waveforms each of a certain amplitude and frequency, while a **pixel** is a mensurated signal, i.e., a signal or signals with some source coordinates attached. A **feature** is simply an enumerated type representing some identifiable portion of a target, while a **symbol** is an enumerated type for the target itself. [69]

Chapter 3

Related Work

A modular or object oriented architecture for sensor fusion has seldom been discussed in the literature, and even less frequently implemented. Hager [37] discusses architecture under the rubric of "system organization," and sees a need for "...library routines ... [which] build an environment that hides irrelevant details and exposes only the relevant concepts that the programmer needs to manipulate." (p. 200) He goes on to suggest that "... the sensor programming environment should facilitate or automate the choice of a sensor system, model, and task description." Henderson, on the other hand, has designed and implemented an architecture based on encapsulated sensor objects in the Multisensor Kernel System (viz., e.g., [46]). His motivations are the same as ours, to wit:

- how to develop a coherent and efficient treatment of the information provided by many sensors, particularly when the sensors are of various kinds;
- how to allow for sensor system reconfiguration, both as a means toward greater tolerance for sensing device failure, and to facilitate future incorporation of additional sensing devices. ([70], p. 82)

The components of his virtual sensor specification are: (1) the logical sensor name, (2) the characteristic output vector, (3) a selector, and (4) alternative subnets. Items 3 and 4 are used to select among other data sources with the same output vector. ([70], p. 85) Henderson's formulation lacks two major elements in our thesis. First, while the "logical sensor" does propagate (or "advertise") a "pedigree," (See [92].) Henderson does not describe how this is computed. He states:

> Thus it would be necessary to classify an algorithm as having a certain degree of accuracy, and, in addition, provide an accuracy function which, given the accuracy of the physical sensor, produces the overall accuracy for the logical sensor which results from the composition of the physical sensor and the algorithm. [70] p. 98

Henderson does not attempt to solve this problem, but merely states:

> ...and more of that [algorithm evaluation] kind of work is required if we are to achieve comprehensive sensor systems.

A number of common sensor fusion algorithms require a sensor specific quantification of the accuracy and reliability of the sensor vector. Second, our architecture supports a virtualization hierarchy, where sensors of a certain type are combined to create a virtual sensor of a different type. Henderson's logical sensor specification does not.

While not explicitly developing a virtual sensor architecture, other researchers have been driven in this direction by their particular problem domain. In [43], for example, the authors describe a distributed fusion blackboard. In addition to the data attributes, they expose a time stamp attribute and a qualitative "confidence measure." This form of 'pedigreeing' supports only three fusion algorithms: weighted averaging, where the confidence measures provide the weights; deciding, where data is prioritized by the

confidence measures; and cueing, where a coarser sensor guides the selection or processing of a finer grained sensor. [34] also describes a hierarchical blackboard architecture, similar to Harmon's, where fusion (i.e. ascension to higher layers in the architecture) is performed by a rule-based hypothesis formulation engine. The intent of the architecture, however, is not sensor fusion but sensor integration and sensor cueing; moving up the hierarchy results in evidential support for hypothesis testing, while moving down the hierarchy cues lower nodes for data specific to the verification or falsification of the hypothesis framed in the parent node.

This is the extent, then, of the current work on sensor virtualization. The scope of the search for related work can and should be extended to include work on decentralized and distributed fusion for the following reasons. First, distributing the data fusion function often entails modularization of the sensor. Second, several of the authors who address distributed fusion also distribute the sensors to the same mobile platforms on which the distributed fusion takes place. In either case they must address the interface problem: what data and what data pedigree to communicate among the distributed elements.

There has been extensive work in the area of distributed Kalman filters. The architecture in [44] assumes local fusion occurs on multiple sensor platforms, which then communicate to a centralized fusion center that performs the global estimate. Brown et al. [11] take a peer-to-peer distributed approach, rather than a centralized approach. Like [44], the authors partition the observation vectors, covariance matrices, and so on, among the distributed platforms, but communicate state error and variance information peer-to-

peer. If we recast this work in terms of sensor virtualization, each mobile platform containing multiple sensors performs its own local fusion, and then contributes the fused estimate, together with a pedigree for the fused estimate, to a second layer fusion algorithm. This architecture meets some of the criteria for a virtual sensor, although it is by no means fusion algorithm independent. In addition, how to determine the error covariance among the platforms remains unsolved. It is this latter issue that is problematic. The key factors in the Kalman Filter are the specification of the covariance matrices and a careful selection of the mean and variance of the noise sources. While the initial error covariance ($\mathbf{P}_0$) is not (generally) important and, in any case, is updated at each time step, determining the system and measurement noise covariances and their covariances ($\mathbf{w}_k$, $\mathbf{v}_k$, $\mathbf{Q}_k$, and $\mathbf{R}_k$) are critical. For each of $\mathbf{w}$ and $\mathbf{v}$ we need some estimate of their standard deviations, and for the matrices $\mathbf{Q}$ and $\mathbf{R}$ we need to be able to determine the covariances:

$$E[\mathbf{w}_i \mathbf{w}^T_j] = \mathbf{Q}_i \delta_{ij}$$

$$E[\mathbf{v}_i \mathbf{v}^T_j] = \mathbf{R}_i \delta_{ij}$$

where δ is the Kronecker delta

Julier and Uhlmann note:

> In many situations the actual statistics are not known perfectly but, providing the information sources are independent, it is possible to "over estimate" the statistics and suboptimal filtering can be performed. However, in many important situations it is impossible to guarantee that the noise sources are independent and might, in fact, be highly correlated. [55]

The authors allude to the common practice of "tuning" the Kalman Filter, e.g., varying $\mathbf{Q}_k$ or $\mathbf{R}_k$ while holding one or the other constant until acceptable performance is achieved. This does not solve the covariance problem, however. In [13], an example of a

diverging distributed Kalman Filter is given where the noise sources **v** and **w** are assumed to be independent.

While we have chosen to introduce the problem of determining measurement covariance by means of the distributed Kalman Filter, it is by no means limited to this particular fusion algorithm. Returning to the fusion examples discussed earlier in this section, we can see an analogous problem appear in each algorithm. Coaxial, homogeneous sensors would tend to have a high degree of correlation, while distributed sensors, or sensor readings from the same sensor at different locations, would tend to be less correlated. One heuristic used by Howard [51][52] in the occupancy grid case is to impose radial sectors around each target, and weigh more heavily range readings taken while the mobile robot is in different sectors. This heuristic has intuitive appeal (and some empirical validation) due to the fact that the sectors impose some lower degree of *a priori* correlation among range readings. Similarly, in the Kalman Filter arena, Grime and Durrant-Whyte [11] maintain correlation data between adjacent nodes in a networked architecture. As pointed out in [13], this is viable only in this restricted architecture.

Maintaining error covariance matrices among distributed sensors is addressed in this dissertation through propagation or generation of the error covariance matrix. As noted in Section 5.2, propagation is by no means always achievable. Generation, on the other hand, assumes that a statistically significant number of measurements is available to correlate, or that reliable a priori information on correlation is at hand. With neither suf-

ficient empirical data nor reliable a priori information on correlation, there is always the potential for the fused estimate to be worse than the best unfused measurement.

To recap, then: some work has been done on modular architectures, but the correlation problem has not really been addressed. In distributed architectures, where the problem is readily evident, only certain distributed Kalman Filters have attempted to address the problem, with varying degrees of success. Finally, it should be noted that the correlation problem of distributed sensors is simply one aspect of the larger problem: that of maintaining, communicating, and correlating some representation of the independence and reliability of the sensor data vector from modular or virtual sensors.

Chapter 4

Sensor Fusion

The two main tasks accomplished in this research are the construction of the virtual sensor specification and the modification or "wrapping" of sensor fusion algorithms to interface with this virtual sensor. These two steps are tightly related: input/output requirements of existing fusion algorithms will dictate the design of the virtual sensor, while the design of the virtual sensor will affect the algorithm wrapper. This section provides an introduction to sensor fusion and to the common algorithms and their interface requirements, and the motivations behind the recent explosion of interest in robotic sensor fusion. This will provide the foundation for the next chapter on sensor virtualization.

4.1 Problem Domain

The single factor that distinguishes the domain of mobile robot research from robotics or even artificial intelligence can best be summed up by the title of J. Borenstein's fundamental work on sensors for mobile robots: "Where Am I?"[8] Mapping, path planning, object manipulation, object recognition all depend on the ability to locate the position and orientation of the robot in relation to the an area of interest in general, or individual objects within this area of interest in particular. Unfortunately, the answer to:

"Where Am I?" is seldom easy to resolve. Even if we assume (as many researchers do) that the initial pose and position are known[4], non-linearities inherent in robot locomotion result in increasing pose and positional uncertainties as the robot moves across the Area of Interest (**AOI**). If we instruct the robot to move from an initial location of (x_{init}, y_{init}), n meters along an azimuth of θ degrees, the new location should be $(x_{init} + n\cos(\theta), y_{init} + n\sin(\theta))$ with a new pose of θ. Given any of a host of electromechanical problems such as friction, wheel slippage, and power levels, the true pose will be $\theta \pm \varepsilon_{azimuth}$, and the new location will be $(x_{init} + n\cos(\theta) \pm \varepsilon_x, y_{init} + n\sin(\theta) \pm \varepsilon_y)$ where $\varepsilon_{azimuth}$, ε_x, and ε_y are error parameters. Not only are $\varepsilon_{azimuth}$, ε_x, and ε_y unknown, but they will also vary in an unknown fashion each time a movement is completed. The resolution of this problem must lie in the relation of the robot to the environment, and in order to determine this relation we must use sensor information.

At this point, we should make a distinction in the problem domain of mobile robots between **navigation** and **localization** problems. The localization problem attempts to determine where the robot is at any point in time, while the navigation problem determines how to pass from some initial point to some goal point. While it might seem at first blush that these two problems are so tightly knit as to defy independent formulation,

---

[4] Some authors conflate *pose* (orientation of an object about one, two, or three axes) and *position* (location of an object relative to some coordinate system). We will keep these terms distinct as they are components of the usual three element mobile robot state vector, i.e., $(x, y, \theta)^T$.

the two problems are commonly treated separately. Partially Observable Markov Decision Processes (POMPD) (see [61][31]) or resolution methods such as STRIPS are navigation formalisms that have been discussed independent of any underlying localization algorithms. By the same token, localization can and has been investigated without reference to any specific navigation task. This latter approach is the one we will take in our research, since sensor science directly only impacts the localization problem.

From the point of view of this work, then, the question "Where am I" is predominantly the localization problem. In order to make this question more tractable, it is useful to break it up into two separate questions. First, after executing the command **move**(n, θ), it is important to be able to estimate $\varepsilon_{azimuth}$, ε_x, and ε_y. This is termed **local localization**. Secondly, it is often useful to determine the current pose and position in relation to the AOI. This is termed **global localization**. These two problems are inter-related, since if the robot pose and position vector is known for some time step k, and $\varepsilon_{azimuth}$, ε_x, and ε_y can be determined precisely, then the pose and position for step $k + 1$ can be derived with the same order of precision. By the same token, if the robot pose and position for consecutive states k and $k + 1$ are known with acceptable precision (i.e., precise global localization), then it is simple to derive the error terms: $\varepsilon_{azimuth}$, ε_x, and ε_y.

There are a number of variations on the theme of localization. For example, the area of interest may be fully mapped in advance, or unmapped. Some researchers (e.g. [36]) assume the former, others, the latter (e.g. [28]). Another variation parenthetically mentioned in [36], is localization with an unknown initial pose and position. All of the

navigation papers can thus be categorized as a three-tuple: mapped or unknown, local localization or global localization, initial point known or unknown. Each combination gives rise to a unique set of problems and the application of a possibly disjoint set of solutions. For example, in [15] Section 4, the problem set is {unmapped, local localization, initial point known}. The author uses a Kalman Filter to improve the pose and position estimate provided by wheel encoders. In [36], the problem set is {mapped, global localization, initial point unknown}. In this case the author uses both Markov Processes and an Extended Kalman Filter to address the global localization problem.

While, as stated above, local localization and global localization are related, they are by no means equivalent. A concrete example of where global localization fails to provide local localization is the problem of **perceptual equivalence classes**. [20] [21] [30] Given a constant set of sensors, this problem describes a situation under which there is more than one location in the AOI that produces the same sensor data. The classical example of this is a robot with sonars at the 12, 3, and 9 o'clock positions. Figure 2 shows two such equivalent states: two identical alcoves at the end of a featureless hallway. In this figure the robot position is indicated by the two large black circles, and the robot pose by the short solid line. In both locations the three sensors will produce the same set of measurements.

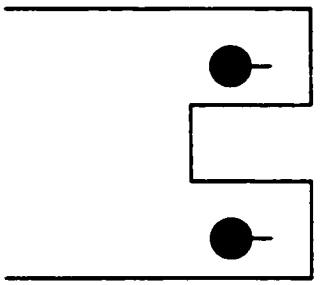

Figure 2: A Perceptual Equivalence Class

This is not merely a strawman argument. One situation where this has occurred in practice is described in [61]. In this experiment the robot takes a set of measurements at some point A, and then receives the command to move down a featureless hallway to some point B. Here a second set of measurements is made. Because the hallway is featureless, both sets of measurements are identical. Since the robot navigates by internally "walking" a state transition graph, and because nodes in the transition graph are distinguished by sensor readings, it is impossible to discover whether the robot has moved or not. This is particularly problematic because one of the most common causes of positional error is wheel slippage. In this case (as the authors discovered) no state transition graph is sufficient to disambiguate the two states.

Because of examples like the previous, it became apparent that navigational algorithms such as the state transition graph cannot be a solution to either the local localization or global localization problem. In most cases, however, improving the quality of the sensor information is. One method for achieving this improvement in quality, and one

that specifically addresses the problems of **precision** and **error** in the sensor measurement, is **sensor fusion**.

For every sensor there is some accuracy measure, which is based on the inherent resolving power of the device. An electromagnetic imager is inherently limited by the wavelength of the portion of the electromagnetic spectrum employed; the lens, number of charge coupled device (CCD) elements, etc. inherently limit the resolving power of a CCD imager; the accuracy of a time of flight (TOF) ranging sensor is inherently limited by the resolution of its timers. Whether or not the measurement precision of the device is sufficient depends on use. The resolution of differential global positioning system (GPS) device is to within 1 meter [8], which is sufficient for locating an aircraft on final approach. This resolution is insufficient for a mobile robot negotiating a crowded room.

In addition to measurement precision, measurement error is also important. Depending on the device, the error may result in sensing a non-existent object, failure to sense an existing object, or producing inaccurate readings. The source of this error may be systemic (component failure, cross talk, etc.), or it may be environmental (specular reflection, signal attenuation, etc.). Once again, given certain applications and certain sensors, this may or may not be problematic.

As pointed out above, depending on the application, measurement error and measurement precision be more or less critical. For those problem domains in which they are, the issue is how to reduce the device error and increase resolution to such a de-

gree that the mobile robot is able to navigation reliably. One method is to equip the robot with sufficiently accurate sensors from the start. Unfortunately, as with most measurement devices, suitably accurate equipment, if available, is frequently unaffordable. The alternative is to make a series of measurements, varying the time, location, or type of sensor, and then to correlate this data in such a fashion as to improve on its accuracy. One set of techniques for achieving this is by sensor fusion.

4.2 Sensor Fusion Algorithms

Algorithms for sensor fusion tend to fall into the following categories: probabilistic and statistical, evidential, fuzzy, and neural nets. Table 1 assigns commonly used fusion algorithms to these categories.[5]

[5] A 1987 study found more than 75 algorithms used for sensor fusion [38]; based on the number of algorithms published since then it is reasonable to assume the current number to be far larger.

- Statistical Methods
  - Kalman Filters
  - Robust Statistics
  - Least Squares
  - Averaging
  - Histogram

- Evidential Methods
  - Bayesian Reasoning
  - Dempster-Shafer Evidential Reasoning
  - Voting
  - Neural Networks
  - Kohonen Nets
  - Hopfield Nets
  - Adaptive Resonance Theory
  - Set Theoretic Filters

Table 1: Common Fusion Algorithms

4.2.1 Concurrent Fusion

Most of the fusion algorithms listed in Table 1 are best suited to fusing co-temporal sensor data, where there are multiple (possible heterogeneous) sensors, presumably registered. Either the measurement error of any one sensor is unacceptable, or it is unknown in advance. In the simplest case of Voting, the majority of sensors in agreement is the value returned by the fusion algorithm. [60] Averaging, on the other hand, assumes confidence in the arithmetic mean, or possibly the mode, of the sensor returns. If there is additional *a priori* information on the reliability of the sensors, the algorithm can take this into account as a weighted vote or weighted average. The algorithm or the weights chosen is governed by some advanced knowledge of the target and sensor characteristics. If these are poorly understood, then an adaptive or learning algorithm such as artificial neural networks [105] or genetic algorithms may be a better choice. Similarly,

evidential algorithms such as Bayesian [47] and Dempster-Shafer [78] update some measure of confidence based on each new measurement.

Often, though measurements are not made concurrently, they are treated as if they were. A robot may make a number of measurements of a target from different locations, and then use a concurrent algorithm to fuse the set of measurements. This is generally used only when the sequence of the measurements and the time between measurements is unimportant, which is generally the case when measured properties of the target do not evolve over time.

4.2.2 Sequential Fusion

One of the underlying assumptions behind concurrent fusion is, assuming the error that infects one sensor's measurements does not infect all identically, that we can employ some algorithm to attenuate this error in the final fused measurement. Taking multiple (sequential) measurements from a single sensor would be a waste of time unless the same assumption was valid: any error that corrupts the sensor measurement changes from time to time and from measurement to measurement.[6] In the simplest case, the properties of the target do not change from moment to moment: the target is motionless, doesn't change size or color, and so on. Any change in the measurement is wholly attributable to

---

[6] In general, a constant measurement error is considered miscalibration, rather than error.

measurement error. Based on knowledge of or assumptions about the error that is corrupting the measurements, we can use various statistical or filtering algorithms from signal processing to extract the uncorrupted signal from the measurement noise. A histogram is the simplest of these techniques [6]. Bandpass, band reject, clipping, and other filters can be used if certain properties of the noise source are known. Moving averages or exponential smoothing are often used if data beyond a certain age is less important than more current data.

Either smoothing or interpolation is very useful if we remove the constraint placed earlier: that the property measured of the target does not change over time. If this constraint is removed then system dynamics comes into play. There are now four parameters to take into account: the target dynamics, noise infecting the target dynamics (such as friction, mechanical slippage, etc., often referred to as "process noise"), measurement of the system dynamics, and noise infecting the measurement of system dynamics (measurement noise). One of the major features of the Kalman Filter is that it takes all four of these into account. [56] Another feature of the Kalman Filter is that it not only performs an interpolation function but also an extrapolation function; based on measurements, system dynamics, and a current estimate of the target, it is able to project ahead one step and estimate the system state at the next measurement period.

Related to the Kalman Filter is the Set Theoretic Filter. In this case nothing is known about the measurement noise or the process noise except that they lie within certain bounds. If these bounds overlap, then the fused estimate is determined to lie within

this intersection, and the error of the fused estimate is bounded by it. A more rigorous explanation of these algorithms is provided in Chapter 5.

4.3 Conclusion

It should be clear from the above discussion that designing a system that performs sensor fusion involves an important choice of algorithm. In addition to the choice being motivated by knowledge of noise and target dynamics, there are additional implementation considerations based on computational complexity, desired accuracy, and general constraints based on the problem domain. This represents one of the major challenges to a theory of sensor virtualization: to construct an architecture that supports any arbitrary choice of fusion algorithm. The next chapter describes how to achieve such a design.

Chapter 5

The Virtual Sensor

The purpose of this chapter is to describe the design of the virtual sensor. There are two parts to this design. The first is the interface, which receives, as input, both control instructions and unfused data from other virtual sensors, and emits fused data. The second is fusion algorithm that resides inside the virtual sensor, which, in some cases, must be modified or "wrapped" to make use of the interface. We begin by providing an initial specification for the interface, similar to that provided in [45] and [46], though modified to support the hierarchy of virtual sensors depicted in Figure 1. We then describe several common fusion algorithms, and show how they the initial specification is insufficient to provide the information necessary for their function. From there the concepts of error and precision are explored, and it is shown why these are necessary parts of the virtual sensor specification. Next the specification is augmented to include error and precision, and the fusion algorithms are modified to make use of the complete interface specification. We then examine how the virtual sensor specification can be extended to support symbol and feature level fusion. Finally, as an illustration, we depict how a virtual sensor can be constructed to support contact sensors.

5.1 Virtual Sensor Interface

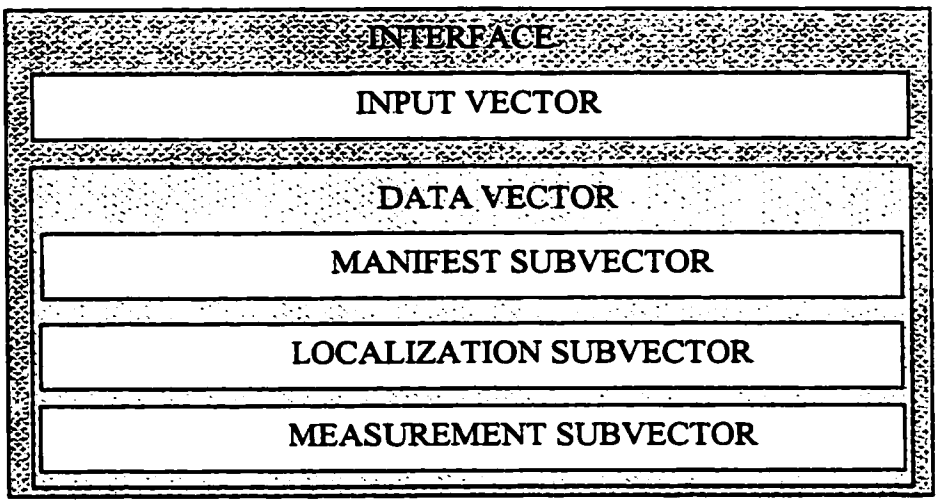

Figure 3: Virtual Sensor Interface

5.1.1 Input Vector

The input or control vector consists of a set of commands and parameters used to control and interrogate the virtual sensor. Some are fairly straightforward whether the interface is to a physical sensor or a virtual sensor. Some are probably only applicable to a physical sensor. In either case, the interface should support a command to interrogate the data vector. What is not so straightforward is whether commands to the physical sensor such as rotation, translation, elevation, etc. should be directed only at the physical sensor, or should be passed from some global sensor management function through and interpreted by each layer of virtual sensors down to the physical sensor. This architec-

tural choice lies more in the realm of sensor management than sensor fusion (e.g., should sensor tasking be centralized or distributed?) however, and need not be solved here. We have chosen to pass the control vector directly to the physical sensor due to simplicity of implementation. Appendices A, B, and C describe the implementation of the control vector for each of the physical sensors developed for the proof of concept.

5.1.2 Data Vector

The data vector is purely a set of output parameters. It is composed of three subvectors: the **manifest, localization** data, and the **measurement subvector**

The manifest contains static descriptive information about the virtual sensor. This may consist of the virtual sensor unique identifier as suggested by [46], software/firmware/hardware versioning information, and a type identifier (ranging, imaging, etc.). This subvector has not been implemented here because it is (1) unproblematic, (2) it is not required for the proof of concept.

The sensor localization subvector contains the pose and positional data for the virtual sensor. Generally the positional data will be a two or three element vector depending on whether the sensor is localized by means of two or three space coordinates. It is an implementation decision whether errors in these parameters should be appended to the localization subvector, or treated as a component in the error parameters of the measurement data vector. We have chosen the latter approach.

The next issue we address in the design of the virtual sensor, is whether and how the localization subvector should be passed to the next layer in a virtual sensor network. In the proof of concept described in Chapter 6 we make use of a vector sonar. (See also Appendix B.) In this device we propagate the localization information through the virtual vector sonar because the measurement vector requires it: range and azimuth data simply make no sense without localization information. We also relay the localization data to perform coverage analysis, as described in Section 5.8.1.

One of the insights that arose in the process of constructing a virtual sensor network was that the localization vector may simply make no sense at layers beyond the physical sensor. If the virtual sensor, for example, is constructed by fusing measurements from two widely separated sonars, there is no location or pose that might reasonably be posited as the virtual location and virtual pose of this virtual sensor. The vector sensor used in the proof of concept (See Appendix B) is a special case. Since the two physical sensors from which it was constructed were close to being coaxial and have the same pose, we simply designated the position of the virtual sensor to be midway between these two sensors, with the same pose. In any case, the localization information is only used for coverage analysis, or to transform measurements that are relative to the location of the physical sensor into absolute coordinates. Thus the localization information is an optional one for virtual (though not physical) sensors.

The final subvector of the data vector is the measurement vector. Because of the restriction to pixel/signal level fusion this consists of quantitative properties: range, ele-

vation, azimuth, absolute target coordinates, temperature, signal strength, mass, and so on. In Section 5.11 we go beyond the proof of concept to describe how to expand this subvector to feature and symbol level virtual sensors. Even in these cases we may want to propagate pixel or signal level measurements.

The measurement vector specification up to this point is essentially the same as that proposed in [45] and [46]. Henderson proposes also to pass a set of "performance features," such as error and precision, but does not explore how these are to be generated. The inclusion of error and precision is essential to the modularization of virtual sensors: to show why this is so, we first have to delve in greater detail into the sensor fusion algorithms.

5.2 Sensor Fusion Algorithms

Table 1 lists a number of commonly used sensor fusion algorithms. Many of these, such as some of the simpler smoothing filters (e.g. median filters, moving averages), and even some of more complex algorithms (e.g. lattice filters, neural nets) require only the actual measurements from the data vector. Thus it is trivial to interface the inputs to a virtual sensor containing these algorithms to the output of another virtual sensor, i.e., wrapping is not necessary. Thus the preliminary specification for the virtual sensor interface provided above will suffice. Three common algorithms, however, require more than just the raw measurements: Evidence Grids, Kalman Filters, and Set Theoretic Filters. Sections 5.2.1 through 5.2.3 introduce the first two plus some common statistical fusion techniques. The last, Set Theoretic Filters, is described in detail in Section 5.8.3.

5.2.1 Evidence Grids

The pioneering work performed by Moravec [76] and Elfes [28] unleashed a flood of researchers exploring and extending the occupancy grid/evidence grid paradigm. The initial concepts were quite simple, and very powerful. The problem domain was mapping an unknown AOI. Presumably, with the aid of some fairly reliable local localization techniques (e.g. odometry), the robot would wander about the AOI sensing its environment, and mapping potential obstacles to its movement. Elfes states:

> This world model [i.e. map] can then serve as a basis for essential operations such as path planning, obstacle avoidance, landmark identification, position and motion estimation, etc. [28]

The second essential element in understanding evidence grids is that they were motivated by the availability of a single ranging sensor: the sonar. With more expensive, more computationally complex, more precise sensors such as vision based, or LIDARs, evidence grids are superfluous. But the sonar, in particular, the Polaroid 6500 module, is less than a perfect source of measurement information. First, the beam width, depending on transducer diameter, is typically 22°. Figure 4 depicts frequency response and beam pattern for the 600 series transducer.

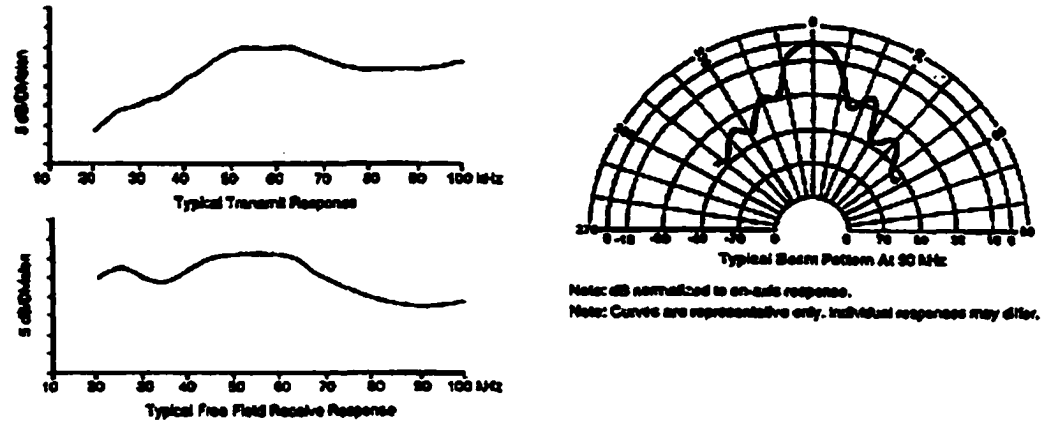

Figure 4: Polaroid 600 Series Transducer [85]

Note that while the documentation claims a range error of ±1 percent, Figure 4 shows quite a different story for azimuth accuracy. This indicates that the transducer is particularly sensitive to reflected energy ±11° (approximately) of the beam centerline. The two side lobes peaking at approximately ±20° are exaggerated by the decibel (logarithmic) scale. So, in general, all other things being equal, more energy from a target located at the beam center will be detected by the transducer than would be from a target located progressively farther from the centerline. But an azimuth uncertainty of ±11° is clearly insufficient for mapping.

In its simplest form, the evidence grid approach begins by tessellating the AOI into $m \times n$ square cells $C_0 \rightarrow C_{mn-1}$. Each cell contains an attribute {occupied} or a pair of attributes {empty, occupied} to which is (are) assigned a numerical value. The robot moves through the AOI taking measurements. Using terminology employed in [62], the **surface hypothesis** is the area bounded by the beam width and the ranging error. The **freespace hypothesis** is the region bounded by the beam width and the surface hypothe-

sis. (See Figure 5.) For each cell within the surface hypothesis, the value of the occupied attribute is increased.

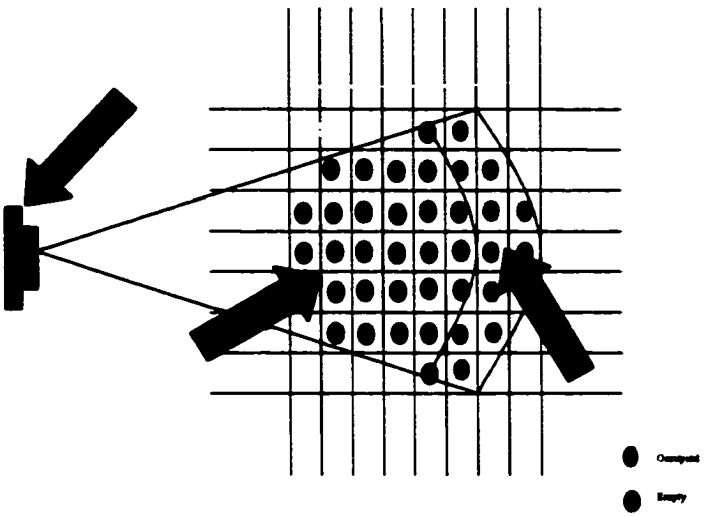

Figure 5: Surface Hypothesis/Freespace Hypothesis

The simplest version of this is Borenstein's histogram approach [6][7], which simply increments the value of the occupancy attribute. The histogram algorithm does not attempt to model the sensor performance, i.e., all cells within the surface hypothesis are treated equally. An early publication by Elfes [28] adds several enhancements. First, he uses the fact that the surface hypothesis represents the first echo return from the sonar pulse to conclude that it is unlikely any cell within the beam width, ahead of the surface hypothesis, is occupied. The sensor model he uses at this point is a parabolic function (which he carefully notes is not, "strictly speaking," a probability density function). The combination of prior fused measurements and new data is similarly ad hoc, i.e.,

$$p_{Empty}(C_i)_{k+1} = p_{Empty}(C_i)_k + p_{Empty}(z_k) - p_{Empty}(z_k) \times p_{Empty}(C_i)_k$$
$$p_{Occupied}(C_i)_{k+1} = p_{Occupied}(C_i)_k + p_{Occupied}(z_k) - p_{Occupied}(z_k) \times p_{Occupied}(C_i)_k$$

Several items of note: First, Elfes initially maintains two attributes per cell: the first corresponding to the assertion that the cell is empty; the second that it is occupied. The values of these attributes are real numbers from (0, -1] in the case of the former, and (0, 1] for the latter. In both cases the value 0 indicates no evidence for the assertion. The second item of note is that the two elements of the methodology in this dissertation are present, although in primitive form: the sensor model and the rule for combining evidence. The sensor model is discussed in Section 5.3.2, we simply note in passing another non-Gaussian model, used in [52]:

$$p(z|C_i) = \begin{cases} 0.05 \text{ if } range(C_i) < z \\ 0.50 \text{ if } range(C_i) > z \\ \frac{\alpha}{z} \quad \text{if } range(C_i) = z \end{cases} \tag{5.1}$$

where α is a normalization constant that yields 1.0 when all cells within the surface hypothesis are summed.

The next step in sophistication of the grid algorithm is an appeal to Bayes' theorem, as first used by Elfes in subsequent to [28], and documented in [29].

One means of specifying the sensor model is as a conditional probability density function of the form

$$p(z \mid x)$$

or the probability of a sensor reading z given state x. Since the immediate objective is to derive x from a set of sensor readings $[z_1, z_2, \ldots z_n]$, we need the reverse of this conditional probability function, or

$$p(x \mid z_1, z_2, \ldots z_n)$$

One version of Bayes' formula is as follows:

$$p(A \mid B) = \frac{p(A)p(B \mid A)}{p(B)}$$

To turn this into an update equation, let

$$p(x_k) = p(A),\ p(x_{k+1}) = p(A \mid B), p\left(z_{k+1}\right) = p(B), p(z_{k+1} \mid x_k) = p(B \mid A),$$

and thus

$$p(x_{k+1}) = \frac{p(x_k)p(z_{k+1} \mid x_k)}{p(z_{k+1})} \tag{5.2}$$

This is the general form of the update equation as used in [29], [51], [52], [62], [81], [73] – in short, all of the evidence researchers with the exception of Elfes' early work, and those using some form of Dempster's rule of combination.

As noted previously, all evidence grid researchers assume that the robot pose and position changes from measurement to measurement. This ameliorates problems resulting from the fact that all measurements are weighed equally, even if multiple measurements are taken from the same location. This is intuitively appealing – because changing sensing locations increases the likelihood that environmental factors effecting measurements in one location will not be present in the next location. (Section 5.3.2.2) Thus several authors [76] [51] [52] incorporate "pose buckets," i.e., radials are drawn from each cell to ensure not more than one measurement from any single target aspect.

Before progressing to the next section where the algorithm is modified to accommodate the virtual sensor, it is important to analyze why the evidence grid algorithms appear to be so effective. As described above the sensor models are extremely simplistic, especially in light of the discussion in Section 5.3.2. Furthermore, the invocation of Bayes' theorem often seems more of a technique for normalizing the sensor model to correspond to an intuitively appealing probability distribution. Nevertheless, the algorithm has proven to be effective in practice for the following reasons.

First of all, if we employ the noise model described in Section 5.4, the distribution of a set of range measurements from a single location is Gaussian[7] with second moment statistics described by **R**. Appealing to the Central Limit Theorems [14], after a large

---

[7] Usually justified with an appeal to the Central Limit Theorem, but see Appendix A for some empirical justification.

number of samples the grid probabilities should mirror the noise model. Thus the sample mean represented by the peak probability in the occupancy grid should correspond to the state estimate.

Second, whether the algorithm incorporates pose buckets or not, all researchers assume that the measurements are taken from different locations. The grid then represents the intersection of multiple surface hypotheses. In short, the algorithm reduces azimuth uncertainty through geometric triangulation.

These first two reasons concern the surface hypothesis solely, and it is no coincidence that they correspond closely to the definition of fusion provided in Chapter 2, i.e., "... is any algorithm that combines commensurate sensor data for the purpose of minimizing measurement error." The third reason for the success of the evidence grid methodologies is the freespace calculation. It serves both a fusion function in that it affects cells previously within the surface hypothesis, and it also performs a coverage analysis function. This latter function will take a central role in the modification of grid algorithms to interface with the virtual sensor, described in the Section 5.8.1.

5.2.2 Statistical Estimation

Assume that there is a set of n concurrent scalar measurements $z_n \in Z$ from a set of n commensurate, registered sensors $s_n \in S$. Any differences among the members of Z can be attributed to calibration errors, process noise, sensor failure, or environmental factors. The simple arithmetic mean of the sample:

$$\bar{z} = \frac{\sum_{i=1}^{n} z_i}{n}$$

can provide an acceptable fused estimate of the system state vector **x** assuming the sample contains few outliers. Extreme data points can be thresholded by computing the sample variance

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^{n} (z_i - \bar{z})^2}{n-1}$$

and rejecting any z_i such that

$$z_i < \hat{\sigma}^2 + k_1 \quad or \quad z_i > \hat{\sigma}^2 + k_2$$

where k_1 and k_2 are some constants derived from the sensor model. Another approach is to use a median filter, which can be less noisy than the average by a factor of 2.5 [27].

If $z_k \in Z$ is a time-ordered series of measurements from the same sensor then we can also use the sample average to compute the state estimate. This suffers from two problems, however. First, since Z is a time series, either the target or the sensor may be in motion. If this series is our sole means of localization, registration and mensuration may be problematic. Secondly, if Z is a very large set, the mean $\bar{z}$ will be very slow to react to rapid changes in the measurements. One method of solving this problem is to use a weighted moving average of the form

$$\hat{x} = f\left(\frac{c_0 z_{k-n} + c_1 z_{k-n+1} \cdots + c_n z_k}{n+1}\right)$$

Selecting a large 'window' n reduces the responsiveness of the estimate, while the selection of various weights (c_i) can be used to exaggerate the contribution of more recent measurements.

All of the algorithms provided above make use only of the measurement data from the virtual sensor data vector, and require no adaptation to interface with it. There are algorithms for which the initial virtual sensor interface specification described at the beginning of this chapter is insufficient. The Kalman Filter is one such algorithm.

5.2.3 Kalman Filters

The Kalman filter, developed by Rudolf Emil Kalman in the early 1960s, is a member of the class of one-step linear mean square estimators.[108] It is an iterative one-step estimator in that, given an estimate of the state of a system at time k, a measurement of some sort that can provide evidence of the system state at k, the first and second moments of random variables representing uncertainty both in the current state and the measurements themselves, then the filter will provide the means to estimate the system state at $k + 1$. The Kalman filter provides for a complex system model and measurement model. These consist of a system state, represented by a vector of size n, a measurement vector of size m, transformation matrices that allow a comparison of measurement vector to state vector, a state vector update matrix, and so on.

While one of the salient points of the Kalman Filter is its ability to operate on vector quantities (vectors of measurements, system state, system input, etc.), the matrix operations complicate basic understanding of its function. So we simplify the filter to scalar quantities in the following explanation, before introducing the common matrix representation of the Kalman filtering equations.

For simplicity's sake, assume that the sensor reading z_1 of a system state x can be modeled as a normal distribution with a mean μ_1 and a standard deviation σ_1. Based on this information alone, the best estimate $\hat{x}$ of x at time k is simply μ_1. Suppose that a second reading is taken with sensor z_2 with a mean μ_2 and a standard deviation σ_2. One method of forming the estimate $\hat{x}_{(k+1)}$ is with the sum of the weighted average of the prior and μ_2, or

$$\begin{aligned}\hat{x}_{(k+1)} &= \frac{\sigma_2^2}{\sigma_1^2+\sigma_2^2}\mu_1 + \frac{\sigma_1^2}{\sigma_1^2+\sigma_2^2}\mu_2 \\ &= \frac{\sigma_2^2}{\sigma_1^2+\sigma_2^2}\hat{x}_{(k)} + \frac{\sigma_1^2}{\sigma_1^2+\sigma_2^2}\mu_2 \\ &= \hat{x}_k + \frac{\sigma_1^2}{\sigma_1^2+\sigma_2^2}(\mu_2 - \hat{x}_k)\end{aligned} \tag{5.3}$$

Letting

$$K = \frac{\sigma_1^2}{\sigma_1^2+\sigma_2^2}$$

results in

$$\hat{x}_{(k+1)} = \hat{x}_k + K(\mu_2 - \hat{x}_k)$$

where K is the "Kalman Filter Gain" and (μ_2 - $\hat{x}_k$) is called the "measurement innovation." This measurement innovation represents the difference between the estimate of x and the measurement of x. The objective, then, is to select K so that the state estimate error variance $P_k = E[(x_k - \hat{x}_k)^2]$ is minimized. Assume that the measurement process can be modeled as a linear discrete Markov process, where v is a zero mean white Gaussian noise sequence, the measurement process can be modeled as:

$$z_k = H_k x_k + v_k$$

and the state process can be modeled as:[8]

$$x_{(k+1)} = \Phi_k x_k + B_k u_k + w_k$$

In the example we are working with here, the state scalar x is simply the range to the target, and since the measurement is the range to the target, the measurement trans-

---

[8] Only discrete time linear models and white Gaussian noise models are described in this dissertation. For continuous and/or non-linear models (and Extended Kalman Filters) see [3]. For other noise models see [3].

formation H from state coordinate space to observation coordinate space is unity. By the same token, since the state is unchanging in this example (neither the sensor nor the target move), the state transition transformation Φ is unity. Neither the input value u nor its transformation are used in this example, so the terms $B_k u_k$ are set to zero. Finally, w is zero mean white Gaussian noise sequence (uncorrelated with v) representing uncertainty in the state transformation process.

Of the many equivalent forms for determining K, one common one is provided below.

$$K_k = \frac{P^-_k}{P^-_k + R_k}$$

where R_k is the measurement error variance, and the *a priori* estimate error variance P^- is given by

$$P^-_k = E[(x_k - \hat{x}^-_k)^2]$$

The Kalman filter is a time-step filter, which consists of predicting the next system state, measuring the next system state, and updating the error variance in the prediction. This process is given by two update equations:

$$\hat{x}_{k+1}^{-} = \hat{x}_k$$
$$P_{k+1}^{-} = P_k + Q_k$$

where Q_k is the process variance.

In addition to the two process equations there are three measurement update equations:

$$K_k = \frac{P_k^{-}}{P_k^{-} + R_k}$$
$$\hat{x}_k = \hat{x}_k^{-} + K_k(z_k - \hat{x}_k^{-})$$
$$P_k = (1 - K_k)P_k^{-}$$

For initial conditions, let $\hat{x}_0 = H_0^{-1}z_0$, i.e., let the initial state estimate be the first observation. As for P_0^{-}, any arbitrary non-zero choice will eventually cause the filter to converge. The sensor model for the Kalman filter, or at least a portion of it, is given by the selection of a value for R_0. The larger the measurement variance (in general), the less vulnerable the filter is to "outliers," and thus, frequently, the greater the estimate variance. Finally, unless the sensor or target moves, the process variance will be zero, so the second process equation simplifies to:

$$P_{(k+1)}^{-} = P_k$$

The discussion above provides a very simplified description of the Kalman filter. Expanding this to the general form of the Kalman Filter requires considering target dynamics, transformation of sensor data to state estimation, addition of a control vector, and the representation of a system as a state vector rather than a scalar quantity. The matrix form of the measurement model becomes

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \tag{5.4}$$

while the matrix form of the state process model becomes

$$\mathbf{x}_{k+1} = \Phi_k \mathbf{x}_k + \mathbf{B}_k \mathbf{u}_k + \mathbf{w}_k \tag{5.5}$$

As for the Kalman Filter equations, the matrix forms for the covariances **P** and **Q** are used rather than variances, Φ becomes a state transition matrix, **H** becomes a measurement transformation matrix, and so on. Augmented appropriately, the five update equations appear as follows:

$$\mathbf{K}_k = \frac{\mathbf{P}_k^- \mathbf{H}_k^T}{\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k}$$
$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-)$$
$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}_k^-$$

$$\hat{\mathbf{x}}_{k+1}^- = \Phi_k \hat{\mathbf{x}}_k + \mathbf{B}_k \mathbf{u}_k$$
$$\mathbf{P}_{k+1}^- = \Phi_k \mathbf{P}_k \Phi_k^T + \mathbf{Q}_k$$

Table 2: Discrete Kalman Filter Update Equations

Figure 6 is a graphical depiction of the iterative nature of the discrete Kalman Filter.

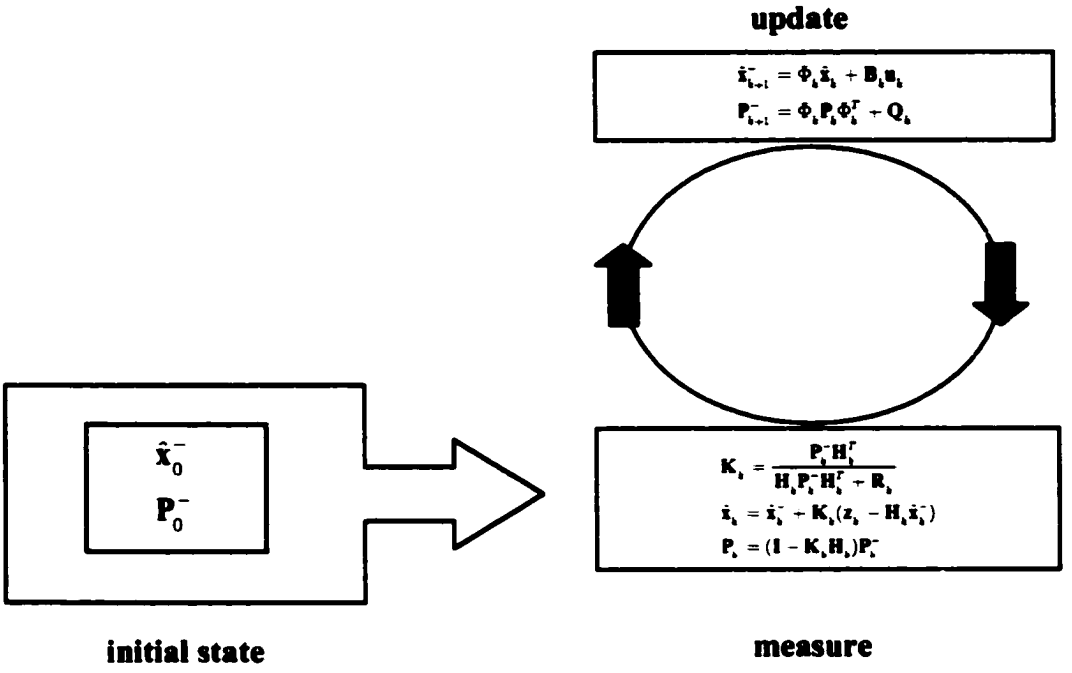

Figure 6: Discrete Kalman Filter Cycle

With some simple algebraic manipulation, substitution of the first equation into the second, and the assignments:

$$W_1 = \frac{P_k^- H_k^T}{H_k P_k^- H_k^T - R_k}$$

$$W_2 = -\frac{R_k}{H_k P_k^- H_k^T - R_k}$$

the second equation in Table 2 becomes the simple linear combination of the estimated state vector and the measurement vector:

$$\hat{\mathbf{x}}_k = \mathbf{W}_1\mathbf{z}_k + \mathbf{W}_2\hat{\mathbf{x}}_k^-$$

i.e., the new state estimate is weighted toward the current state estimate if the state estimate covariance is low, or toward the measurement vector if the measurement noise covariance is low. Note that this is the matrix form of the weighted estimate provided in equation (5.3).

In the forty years since the emergence of the Kalman Filter, numerous modifications have been introduced. They have been extended to continuous time, correlated signal and noise, non-white noise, and non-linear state transformations.[33] The latter case, called the Extended Kalman Filter, replaces the measurement model with

$$\mathbf{z}_k = f(\mathbf{x}_k, \mathbf{v}_k)$$
$$\mathbf{x}_{k+1} = g(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k)$$

where $g(.)$ and $f(.)$ are non-linear equations.

Some observations on the Kalman Filter: First, assuming that the observations are probabilistically independent, or the degree of dependence is known and determined by a cross covariance matrix, it can be relied upon to produce a fused estimate with an error less than either estimate alone. Second, while it relies on data that might, at best, be em-

pirically supported (e.g. noise covariance, estimates of initial conditions), it is extremely robust in the face of errors in these estimates, and will, in most cases, converge.

5.3 Error and Precision[9]

The preceding discussion of the Kalman Filter provides some initial insight into the inadequacy of the initial virtual sensor interface specification described at the beginning of this Chapter: it has no provision for conveying measurement error. What will be proved in the following sections is the more general claim: the initial specification and, therefore, the "logical sensor specification" described in [45] and [46], are not sufficient to support the sort of modularization required of a virtual sensor. Two parameters missing from the initial interface are sensor error and precision.

5.3.1 Error

A fundamental factor in sensor data is sensor error. Error asks the question: "Assuming that sensor precision is known and accounted for, how accurate can I assume the measurements to be?" The error model of a sensor may be the composite of many factors. First, there are environmental factors. One of the problems of sonars in highly specular environments is that target composition can also result in measurement error. Sonar measurement reliability is also affected by temperature, air pressure, humidity, at-

---

[9] Realizing that **precision** has an accepted definition in statistics relating to the confidence interval for a random sample, it is not used here unguardedly. Other appropriate terms such as **resolution** are similarly rich in connotation (see [38]), and the Thesaurus provides no adequate alternatives.

tenuation, and scattering. Finally, there are intrinsic reliability factors of the sensor itself: cross talk between electrical components, thermal noise, beam width (for directional sensors), calibration, bandwidth, device failure, and so on. Since absolute reliability is unachievable, the next best thing is an accurate assessment of the reliability of a measurement under a specific set of test conditions.

In surveying the literature on sensor fusion, we have found two distinct models of error. First: the **sensor model**, is most often used when the fusion algorithm is evidentiary in nature. Second: the **noise model**, is more often used with statistical fusion algorithms.

5.3.2 Sensor Model

The connection between Bayesian inference methods and the sensor model is no clearer explained than in the following by Judea Pearl:

> Bayesian methods require the specification of a complete probabilistic model that relates the set of hypotheses to the set of anticipated observations. [83, p. 540]

This relation is typically described as the conditional probability $p(z \mid x)$ where x is the hypothesis and z is an anticipated observation. There are three counterparts to this expression that can be derived by forming the complement of the observation and of the hypothesis.

$$\begin{aligned} &p\left(z \mid \bar{x}\right): \text{False positive} \\ &p\left(\bar{z} \mid x\right): \text{False negative} \\ &p(\bar{z} \mid \bar{x}): \text{ not germaine in this context} \end{aligned} \tag{5.6}$$

The complements in equation (5.6) can be made clearer by expanding the original expression to: $p(z = c \mid x$ is the case) where c is some constant. For example, with a ranging sensor, $z = c$ means that the measured range z returns the value c, which happens to be the range to target x. Thus the false positive (or Type 1 error) is the probability function where the measurement z returns the expected parameter c but the hypothesis is false, while the false negative (or Type 2 error) is the probability function where the hypothesis is true but the measurement indicates otherwise.

To illustrate this model, consider the following specific example of occupancy or evidence grids.

5.3.2.1 Error Sources: Sensor Error

Ideally, the sensor model would be:

$$p(r \mid D) = \delta_{r=D} \tag{5.7}$$

and the notional occupancy probability would be:

$$p(C_i \mid r) = \frac{1}{2}\delta_{r-D_i} + \frac{1}{2}u_{r-D_i} \quad [10]$$

where δ is the unit impulse function and u is the unit step function. In fact, in the ideal case depicted in equation (5.7), the occupancy grid may be dispensed with altogether since all measurements are error-free.

Key to the Bayesian (and pseudo-Bayesian) update algorithms

$$p(C_i \mid r) = \frac{p(r \mid C_i)p(C_i)}{p(r \mid C_i)p(C_i) + p(r \mid \overline{C_i})p(\overline{C_i})}$$

is the term $p(r \mid C_i)$, which is generally computed from the sensor model. [28] employed a model (non-probability distribution function) of the sort

$$p(C_i) = O_r(D) * O_a(\varphi)$$

where (using Elfes' notation) $\varphi = \Omega - \theta$ (the angle between the sensor azimuth and the target azimuth) and

---

[10] Note: C_i is shorthand for the hypothesis "C_i is occupied," while r is shorthand for "$z = c$" where c is some constant.

$O_r(D) = 1 - ((D - r)/\varepsilon)$ for $r - \varepsilon \leq D \leq r + \varepsilon$, 0 otherwise

$O_a(\varphi) = 1 - (2\varphi/\phi)^2$, for $-\phi/2 \leq \varphi \leq \phi/2$

In this model, the probability of detection is seen as a function of the Cartesian distance between a point lying at the intersection of the beam azimuth and the detected range. The two sensor properties that are modeled are range error (ε) and beam width (ϕ).

In a later formulation [29], range error is no longer modeled by a additive scalar, but by Gaussian distributions representing "the normal behavior of the sensor" (presumably ε), and "occasional gross errors" (perhaps specular reflections). [62] modifies the standard deviation of the Gaussian distribution to account for range induced error (the greater the range, the greater the measurement error) and atmospheric absorption (the greater the range, the weaker the return echo). Note that this entertains both a sensor model (represented by the Gaussian distribution) and an environment model (represented by the atmospheric absorption). This sensor model is partially justified by the 6500 data sheets [86], while the environment model is presented in [62] with no obvious justification (other than intuitive plausibility).

One property of the 600 series electrostatic sonar transducer that is not fully addressed by any of the sensor models reviewed is the beam pattern itself. (Figure 4.) Many model the beam width as a 25°-30° cone, which matches a 30db drop in the typical beam pattern. Note first that this is nominal: each transducer may vary. Secondly, since

the majority of the energy illuminating the target is centered on the beam azimuth, the probability of detection (*ceteris paribus*) is greater the closer the target is to the beam azimuth. [62] provides as "a first approximation" a Gaussian expression with the mean equal to the sensor azimuth and the standard deviation equal to the beam width/2.

5.3.2.2 Error Sources: Environmental

In [17] the author makes the general claim that one of the deficiencies in other researchers' sensor model formulations is their failure to take into account the "environment." This claim is incorrect in the case of [62] where distance related range attenuation is modeled. The author's case is further weakened by his own inability to distinguish environmental factors (e.g. ambient air temperature) from target properties ("reflection coefficient of obstacles"). However, the general point made by the author concerning the importance of sophisticated error models is an important one.

The occupancy grid community shows an evolution toward more and more sophisticated (i.e., complex) sensor models, with [62] representing the most complete generic model, and [17] representing (potentially) the most accurate tailored model. In a general theory of sensor modeling, the error model would take into account scattering, attenuation, propagation, and other factors. Some (such as attenuation) would affect probability of detection in the error model, some (such as the density of the media) would affect range errors in the error model. Given the range limitations of this particular sensor, and the nominal operating environment, which of these would have a discernible impact on the error model? Clearly, indoors, and at ranges less than 10 meters, the environment is sufficiently stable and predictable that it probably is not a significant con-

tributor to error. [62] models signal attenuation based on distance, which is probably unwarranted since any decrease in the probability of detection due to distance is more likely a result of the target cross-section than signal attenuation (especially at close range). This is due to the fact that the 6500 ranging module that drives the 600 series transducer automatically compensates for attenuation.[11] As for changes in air temperature, the speed of sound is related to temperature by the following:

$$c \cong 33{,}145 + 61(T\text{°C}) \text{ (cm/sec)} \quad [2]$$

While indoor temperatures seldom vary more than a few degrees centigrade during the course of an experiment, one can expect ranging errors due to temperature fluctuation to be as much as six centimeters at extreme ranges, and thus must be factored into the range estimate.

Other environmental influences on signal attenuation described in [2] are: humidity, temperature, and frequency. Once again, neither humidity nor temperature appear to

[11] There is a stepwise gain produced by the SN28784 over a period of 38ms to a maximum of 10, which compensates for signal attenuation.

be a significant factor in the environment model given an indoor setting and the short ranges allowed by the 6500 module, and frequency fixed by the transducer at 50kHz.[12]

Brown distinguishes uncorrelated noise (e.g. timing errors) that can be modeled by white noise with a normal probability distribution function, from correlated noise (e.g. environmental effects). He states that the time of flight for any partition is uncorrelated to any other due to temperature/humidity effects. ([10], p. 197) In the case of feature extraction using sonar arrays such as described in Appendix B, Brown finds that the surface normal is insensitive to environmental affects. (i.e., the relation between the angle of incidence of the sonar beam to the angle of reflection is unaffected.) Thus, even if environmental factors result in ranging errors, the environment model can be disregarded in certain instances of feature extraction.[13]

5.3.3 Error Sources: Target

The final part of the error model is the target model. In order to construct a target model the target must first be identified. Predictive algorithms such as the Kalman filter and some target tracking algorithms implicitly assume that the target has been identified, and thus the target behavior is predictable. None of the evidence grid papers identify, model, or accommodate target properties in the error model.

---

[12] There are applications were greater accuracy is required, and thus environmental factors do play a part. In [59], for example, the environment model is required to achieve a range accuracy of 1mm.

[13] Supported by [93] p. 169.

From a physical perspective, target error for an ultrasonic ranging sensor would include such things as: target pose/position, target geometry, surface roughness, and acoustical damping (acoustic impedance). The first three affect such things as echo scattering; the last affects the amplitude of the returned echo. These affect the error model in the form of specular reflection and probability of detection.

The value of target recognition (and thus, the error model) to the general mapping problem cannot be overemphasized. Object recognition can provide a far more reliable method of removing erroneous sensor data than simple thresholding (to which occupancy grids ultimately reduce). For example, a number of the occupancy grid papers (e.g. [22], [28], [29], [36], [51], [52], and [62]) provide data from sample runs in regions consisting of wall objects, corner objects, and door objects. Since corners and door openings are notorious for producing specular sonar responses, recognizing that a set of measurement data is produced by a corner or a door makes it easier to filter erroneous measurements.

5.4 Noise Model

While the sensor model has its origins in evidential reasoning, the noise model belongs to the foundations of signal processing. The concept is straightforward: a measurement begins with a set of properties of an object or a process we wish to measure. A noise term is identified for each property: commonly these are modeled as independent, zero mean random variables, but this need not be the case. The measurement, then, consists of the arithmetic sum of the each property with its corresponding noise term.

As an example, suppose that that we wish to measure a single property: the voltage from a direct current power supply. This voltage is represented by the state variable x. Since this is direct current, the state variable is constant across time, i.e., $x_k = x_l$ for any k and l. Under ideal circumstances, the measurement of the voltage taken of x at k (z_k) would be exactly equal to x_k. The measurement of x_k, however, is corrupted by other signals such as a 60 cycle AC hum from a fluorescent light in which the signal is being measured. This is represented as additive to the state variable, i.e.,

$$z_k = x_k + v_k \tag{5.8}$$

where v_k is the voltage induced into the power supply by the florescent light at time k. There are several things to note about equation (5.8). First, v is not a direct current – since it is alternating current, the voltage varies across time. Secondly, z_k is now in error by exactly v_k. Figure 7 depicts the original 5.0 volt DC voltage v_k (dotted line), the 1.0 volt AC hum v_k (bottom sine wave), and the measured voltage z_k (top sine wave).

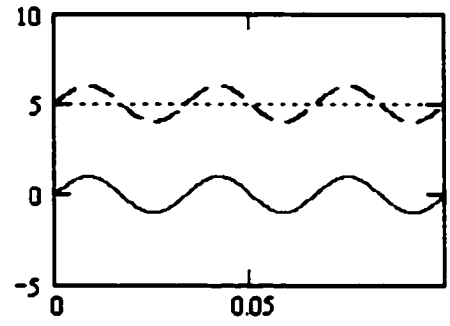

Figure 7: DC Voltage with 60 Cycle Ripple

As a computational and practical matter, the above example is unproblematic. To recover x_k from z_k it is only necessary to subtract v_k. In circuit design one commonly inserts a filter: either a band reject filter tuned to 60 cycles or a voltage regulator that smoothes the ripple induced by v.

Next, assume that v is not one signal, but many different signals. v could be decomposed into individual signals by a number of techniques, such as Fourier analysis. If the composition of v is not known in advance, or as a simplifying assumption, it is common to represent it as a zero mean random variable with a normal (Gaussian) distribution. One motivation for this is that a frequent component of measurement error, thermal noise, can be accurately modeled by a Gaussian process [33]. Secondly, due to the Central Limit Theorem, if each v is considered to be the sum of a large number of mutually independent random variables, then the distribution of v is normal. [14][75] Third, con-

sidering the system noise term v in equation (5.8) as independent, zero mean, and normally distributed results in certain statistical simplifications. For example, as a Gaussian distribution, v can be completely characterized in terms of its first and second statistical moments. Also, if the system state x is constant, or itself a normally distributed Gaussian random process and x and v are independent and thus jointly Gaussian, then the measurement z will also be a Gaussian random variable. [33][75] Figure 8 depicts the original 5.0 volt DC signal, this time corrupted with zero mean Gaussian noise with a 0.5 volt standard deviation.

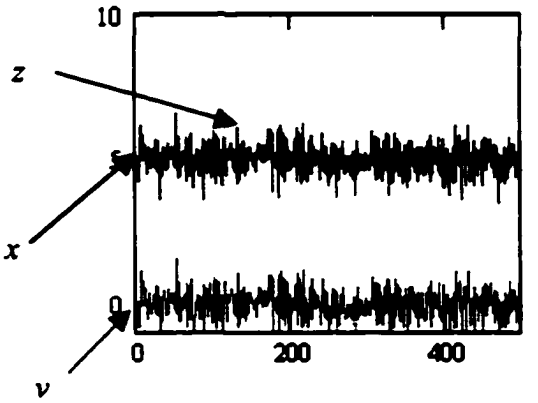

Figure 8: DC Voltage Corrupted with Noise

Finally, equation (5.8) is easily generalized to a case where **z**, **x**, or **v** are vectors, not scalars, and the size of **z** may be different from **x**. This may be the case where certain properties of **x** are measured, while other properties are inferred, or not measured at all. The $m \times n$ measurement matrix **H** (where **x** is a vector of dimension n, **v** is of dimension

m, and **z** is a vector of dimension *n*) is used to transform the state vector **x** into the measurement vector **z**. Thus equation (5.8) becomes the general measurement model:[14]

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \tag{5.9}$$

5.5 Sensor Model vs. Noise Model

The differences between these two models of error are profound. The sensor model describes error in terms of false positives and false negatives, i.e., equation (5.8). In the Boolean logic of complementary possibilities, there is no simple method for accounting for magnitude of error: either detection occurs or it doesn't. The noise model (the vector $\mathbf{v}_k$ in equation (5.9)), on the other hand, computes the error magnitude as a distribution, but contains no means for explicitly expressing false positives and false negatives.

A second difference between the two models is that the probability of detection model is more general; the state vector **x** in the measurement model (equation (5.6)) need not be related to the measurement vector **z** by a set of deterministic transformations. All that is required is some (often poorly justified) conditional probability density function. For example, let *x* be the hypothesis "The target is a helicopter," and *z* be the measured velocity of the target. If *z* = 700kts., then *p*(*z* | *x* is a helicopter) is close to 0.0. This gen-

[14] For an overview and a taxonomy of the various colors of noise, see [77].

eral nature of the model comes at a price, however: it often conceals complexity. Return to the occupancy grid example provided above for a moment. Recall that the form of the sensor model for the occupancy grid is $p(z \mid C_i)$, i.e., the conditional probability of a certain measurement z given the fact that cell C_i is occupied. C_i is simply shorthand for the state vector $[x, y]$. By the same token z is simply shorthand for the measurement vector $[r, \phi]$. The probability density function is really the joint probability $p(z_1 = r)$ and $p(z_2 = \phi)$, where

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} r \\ \phi \end{bmatrix}$$

So the simple conditional probability expressed in equation (5.6) becomes a much more complex union of conditional probabilities if both terms are vectors. For example, the false negative in that equation expands to

$$p(\bar{z} \mid x) = p(\bar{\phi}, r \mid x) \cup p(\phi, \bar{r} \mid x) \cup p(\bar{\phi}, \bar{r} \mid x)$$

if z is a two element vector (azimuth and range). Another complicating factor in this expansion is whether azimuth and range are independent. All of the occupancy grid papers we have reviewed assume that this is the case, which may a valid assumption for the sensors they use. As described in Appendix B, however, this is not true for all sensors.

5.6 Error Models and the Virtual Sensor: Practical Issues

It is clear from the discussion in Section 5.3 that any function generating an error metric for a virtual sensor must take into account all three sources of error: Sensor, Envi-

ronment, and Target. The sensor error model can be maintained quite successfully within the virtual sensor, and presented to the interface layer as an independent parameter. The environment error and, even more so, the target error, require information not easily derivable from within the virtual sensor. For example, suppose that the environment contains not just the target but a number of other reflecting surfaces. This "clutter" will result in specular reflections, which will cause measurements to be in error. Thus, ideally, the sensor error parameter should be increased in proportion to clutter. Many sensors, however, have no means of determining the extent of the clutter.

The target error may also be a problem since, unless the sensor is a symbol level sensor [69], it may not contain a target model. If there is no target model, then it is difficult to generate a target error parameter.

To return to the virtual sensor specification, we have chosen, as a first approximation, to maintain the sensor model and relevant portions of the environment model in the virtual sensor. The error parameters will be included in the virtual sensor interface data vector (see Figure 3) as a measurement error covariance matrix. (Figure 9)

$$\begin{bmatrix} \sigma^2_{\varepsilon_1} & K_{\varepsilon_1\varepsilon_2} & \cdots & K_{\varepsilon_1\varepsilon_n} \\ K_{\varepsilon_2\varepsilon_1} & \sigma^2_{\varepsilon_2} & \cdots & K_{\varepsilon_2\varepsilon_n} \\ \vdots & \vdots & \ddots & \vdots \\ K_{\varepsilon_n\varepsilon_1} & K_{\varepsilon_n\varepsilon_2} & \cdots & \sigma^2_{\varepsilon_n} \end{bmatrix} .$$

Figure 9: Measurement Error Covariance Matrix

5.7 Precision

In describing the precision of the sensor measurement, the sensor measurement is asserted to be correct to within some limit of the accuracy of the measuring device. For example, for a sonar that uses software to time the return echo, if the software loop takes 30 μs, the sonar resolution is limited to 5 mm. From the manufacturer documentation and our own tests we know that the resolution of the Milford Sonar (see Appendix G) is 10mm, due either to the Scenix embedded processor timer or the embedded software controlling it. The BX-24 embedded processor's timer that we use in the proof of concept is far more precise, allowing resolution of approximately 0.0688mm. (See Appendix A.)

As the above example shows it is clear that even the same sensor types (sonar) may have different orders of precision. Even with identical sensors, individual characteristics of the sensor as determined through calibration may result different precision --- the less sensitive the sensor, the less precise it becomes. The sensor or power source may degrade during use, and thus lead to individual differences in precision among sensors that are, in all other respects, identical. Precision, like error, may have to be continually updated by the virtual sensor.

So sensors may have differing precision. The impact of not factoring precision into the sensor model can be seen from the following example.

Suppose that S_1 and S_2 are identically performing sensors. Each returns a measurement of the location of a target in terms of its x and y coordinates. Using the symbology developed in equation (5.9), the true location of the target will be represented by the vector $\mathbf{x}$, while the measurement vector returned by S_1 and S_2 will be represented by the vectors $\mathbf{z}_1$ and $\mathbf{z}_2$, respectively. The target location will be assumed to lie at the midpoint between $\mathbf{z}_1$ and $\mathbf{z}_2$. The measurement error will be modeled as a zero mean, Gaussian random variable. Since these are identical performing sensors, the standard deviations for each on the x axis and y axis are identical. Since the sensors are identical, the Kalman Filter will produce an estimate of the location of $\mathbf{x}$ ($\hat{\mathbf{x}}$), as the midpoint between $\mathbf{z}_1$ and $\mathbf{z}_2$. So the error, which is computed as the difference between the estimated target position and the real target position, is 0.

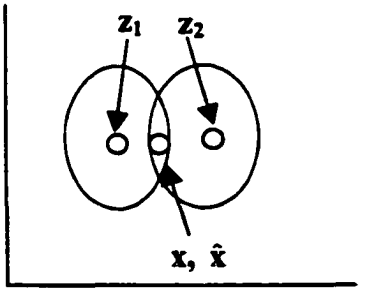

Figure 10: Identical Sensors

Figure 10 shows the situation described above, with the error as an elliptical contour surrounding each measurement.

Next, suppose that S_1 is as described above. The measurement from S_2 is thresholded, however, so all that measurements from the S_2 sensor lie within its precision

boundary, regardless of how erroneous. (This is roughly analogous to the comparison between the Milford Sonar and the proof of concept sonar as described previously.) Because of this, S_2 shows no measurement error. However, when we once more pass these measurements through the Kalman Filter, z_2 is eliminated. It is important to notice that, because of the greater imprecision of S_2, and because the Kalman Filter performs a linear weighting of the measurements' contribution to the fused result by their variances, S_2's measurement (z_2) is essentially discarded. The resulting estimated target location is depicted in Figure 11, and the error of the fused estimate is now greater than 0. Because we know that S2's imprecision is concealing a Gaussian error distribution, we could have modeled the imprecision of S2 as error, and arrived at the same estimate as in Figure 10. But this is an unusual case. We have a very similar, much more precise sonar, and we can hypothesize a Gaussian error model being hidden by the imprecision. More often there is the case like the laser range finder described in Appendix C, where we know only the bounds of precision, and not the statistical distribution of the measurement within these bounds. The Kalman Filter requires the error distribution to be known.

So in the example above the Kalman Filter dispenses with all measurements from S_2, and the resulting fused estimate is the same as the unfused measurement from S_1. There are a set of algorithms that can make use of S2, however, and improve on the estimate provided by the standard Kalman Filter by making use of precision. One of these mixed statistical/set theoretic filters is mentioned in Section 5.8.4, and we use a simplified form in our proof of concept.

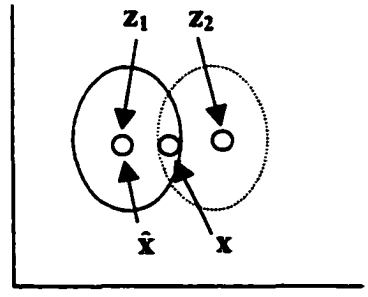

Figure 11: Fused Estimate

There are a number of ways to incorporate precision into the sensor vector and the algorithmic wrapper. The two that will be examined here are the most felicitous to the sensor virtualization/algorithm wrapping.

The simplest is to match the discrete measurement intervals to the grossest level of precision. For example, if S_1 is precise to 1 mm, while S_2 is precise to 1cm, reduce the precision of S_1 to S_2, so that a measurement between 0.0 cm through 0.9 cm in S1 is mapped to 0.0 cm; 1.0 cm through 1.9 cm is mapped to 1.0 cm, and so on. The algorithm would be as follows: (1) from among the measurements to be fused, select the sensor with the least precision. (2) Create a many-one function for all other sensors to map their measurements onto a single set of discrete values. This process has the effect of "fuzzy-ing" all measurements, i.e., reducing measurement resolution for all sensors down to the lowest common denominator. While this is certainly the simplest approach, there will be a loss of precision in the fused measurement.

A second approach is to describe the system state x as an unknown bounded by the precision term:

$$z - \xi \leq x \leq z + \xi$$

and then to treat x as a random variable with some probability distribution. This immediately leads to the question of how to characterize the probability density function for precision. Prior to empirical verification, it is reasonable to assume that for any particular sensor, any one of an infinite number of probability density functions might be the case. Two distributions are particularly attractive for intuitive and formal reasons: the uniform probability density function and the Gaussian probability density function. [14]

The Uniform Distribution, lacking any empirical evidence to believe otherwise, has a great deal of appeal. It basically says that, given a measurement z, the true value is equi-probably distributed along the continuum between z-ξ and z+ξ. The density function for such a distribution is given in equation (5.10).

$$\begin{aligned} p(x) &= \frac{1}{N}, \quad for \quad z - \xi \leq x \leq z + \xi \\ p(x) &= 0, \quad otherwise \end{aligned} \tag{5.10}$$

The distribution function is given in equation (5.11).

$$F(x) = \begin{cases} 0, & for \quad x < z - \xi \\ \dfrac{x - (z - \xi)}{2\xi}, & for\ z - \xi \le x \le z + \xi \\ 1, & for \quad x > z + \xi \end{cases} \tag{5.11}$$

The uniform distribution has been used by some authors for the surface hypothesis in the evidence grid. For example [81] distributes the probability that a cell is occupied uniformly across the surface hypothesis thusly:

$$P_F(i,j) = \frac{1}{N} \quad \text{for all cells } (i,j) \text{ in the surface hypothesis}$$

where N is the number of cells in the surface hypothesis.

We do not endorse this approach when a higher fidelity sensor model is available, for example, the Gaussian model, where the probability that a cell is occupied is greater for those cells along the center line of the beam. We do, however, use this distribution for mapping bounded-but-unknown distributions to the evidence grid in the proof of concept.

Another possible distribution for representing precision is the normal, or Gaussian distribution, the density function for which appears in equation (5.12).

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)/2\sigma^2} \tag{5.12}$$

There is a natural interpretation for precision where a density function is available for computing measurement error. The computation of a probability for a continuous random variable X over a closed interval $[a, b]$ from a density function $f(x)$ is as follows:

$$\Pr[a \leq X \leq b] = \int_a^b f(x)\,dx \tag{5.13}$$

By equating $a = z - \xi$ and $b = z + \xi$, equation (5.13) can be used to model the sensor probability of detection for a measurement z bounded by precision ξ. This interpretation of precision will be used in Section 5.8.1 to "wrap" the Bayesian evidence grid fusion algorithm.

One alternative approach to the distributions discussed above is to represent precision simply as bounded-but-unknown. This is not appropriate for some fusion algorithms such as evidence grids or Kalman Filters. In Section 5.8.3 we describe one type of fusion algorithm, the set theoretic filter, for which this interpretation is appropriate.

Returning to the virtual sensor specification, we must represent precision in the measurement vector. For any measurement z, there is an associated precision constant ξ. For a measurement vector $\mathbf{z}$, we choose to represent precision in matrix form as:

$$\boldsymbol{\xi} = \begin{bmatrix} \xi_0^2 & 0 & 0 & 0 \\ 0 & \xi_1^2 & 0 & 0 \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & \xi_{n-1}^2 \end{bmatrix}$$

such that $\Omega_\xi=[\mathbf{x} : \mathbf{x}^T\xi^{-1}\mathbf{x} \leq 1.0]$ is an ellipsoidal set in n space. The reason for this choice is discussed in Section 5.8.3. It should also be noted that, through propagation of precision, the off-diagonal elements of the precision matrix may no longer be zero.

5.8 Algorithm Wrapping

Now that the initial virtual sensor interface specification has been augmented to allow for error and precision, the following sections describe whether and how common fusion algorithms are "wrapped" to take advantage of this.

5.8.1 Evidence Grids and the Virtual Sensor

The first step in "wrapping" evidence grids is the surface hypothesis calculation. Given the virtual sensor specification described above, two sensor models can be accommodated: the Gaussian distribution by means of the error matrix[15] and a bounded-but-unknown distribution by means of the precision matrix. Other distributions require communicating the model in the data vector, as is described in Chapter 7, or by **generation**. Generation, it may be recalled, is where the virtual sensor at layer $n-1$ constructs information concerning the virtual sensor at layer n based on a priori information of the characteristics of the layer n virtual sensor.

As noted above, most authors model both azimuth and range for sonars as a bivariate normal distribution, and thus the range and azimuth variances are passed as the

---

[15] By making an explicit assumption that the error is normally distributed, and therefore the error matrix represents the statistical moments of a normal distribution.

traces of the error matrix. All assume independence, which is probably a benign assumption in the case of the sonar model. Equation (5.14) is a common sensor model for the Polaroid sonar (See, e.g. [24], [29], and [62]), where the difference between the measured azimuth and range and the cell azimuth and range take the place of the difference between the random variable and its mean used in the standard bivariate function.

$$f(c_i = occ | r_{cell}, r, \phi_{cell}, \phi) = \frac{1}{2\pi\sigma_r\sigma_\phi} \exp\left(-\frac{1}{2}\left(\frac{(r_{cell} - r)^2}{\sigma_r^2} + \frac{(\phi - \phi_{cell})^2}{\sigma_\phi^2}\right)\right) \quad (5.14)$$

where r is the measured range, r_{cell} is the distance to cell c_i, ϕ is the target azimuth, ϕ_{cell} is the azimuth of cell c_i, and σ_r and σ_ϕ are the standard deviations of the range measurement and azimuth measurement, respectively.

However, as pointed out in Appendix B, some sensors have a non-zero covariance. So in order to generalize the sensor model, the first step is to include the measurement covariance ρ, which is the covariance of the range and azimuth measurements.

$$f(c_i = occ | r_{cell}, r, \phi_{cell}, \phi) =$$
$$\frac{1}{2\pi\sigma_r\sigma_\phi\sqrt{1-\rho^2}} \exp\left(-\frac{1}{2}\left(\frac{(r_{cell} - r)^2}{\sigma_r^2} + \frac{(\phi - \phi_{cell})^2}{\sigma_\phi^2} - 2\rho\left(\frac{(r_{cell} - r)}{\sigma_r}\right)\left(\frac{(\phi - \phi_{cell})}{\sigma_\phi}\right)\right)\right) \quad (5.15)$$

where

$$\rho = \frac{K_{r,\phi}}{\sigma_r\sigma_\phi}$$

This equation also shows how the correlation coefficient ρ, required by equation (5.15), is derivable from the error covariances K contained in the error covariance matrix of the virtual sensor. (See Figure 9.)

The next step is to consider precision. Oddly enough, both Konolige and Elfes seem to confuse probability density functions with probabilities. In Konolige's case this is obscured by quickly computing likelihood ratios, which avoids the embarrassment of a "probability" ≥ 1.0 when the azimuth standard deviation is less than or equal to $\frac{1}{2\pi\sigma_r}$. Precision provides a simple solution to this problem: probabilities are calculated from the probability density function by integrating the area within the bounds of the precision terms.

$$p(c_i = occ \mid r_{cell}, r, \phi_{cell}, \phi, \xi_r, \xi_\phi) = \int_{r_{cell}-(r_{cell}\%\xi_r)}^{r_{cell}-(r_{cell}\%\xi_r)+\xi_r} \int_{\phi_{cell}-(\phi_{cell}\%\xi_\phi)}^{\phi_{cell}-(\phi_{cell}\%\xi_\phi)+\xi_\phi} f(r_{cell}, r, \phi_{cell}, \phi) d\phi dr \qquad (5.16)$$

where $f(r_{cell}, r, \phi_{cell}, \phi) =$ equation (5.15)

In order to clarify the complicated bounds of integration in equation (5.16), recall that the purpose of this equation is to compute the probability that cell c_i is occupied given a range measurement of r and a measured azimuth of ϕ. Because of the imprecision ξ_r and ξ_ϕ, a measurement of r and ϕ is really the measurement $r \pm \xi_r$ and $\phi \pm \xi_\phi$. By the definition of precision, we can be sure that r is an integer multiple of ξ_r, and ϕ is an

integer multiple of ξ_ϕ. In other words, if ξ_r = 1cm, then we can be certain that r = 1cm, 2cm, etc., but not 1.1 cm. This is not the case with r_{cell}, and ϕ_{cell}, however, since the true (not measured) angle and range of the cell from the sensor is not a function of the measurement precision. Because of this the bounds of integration are adjusted so that $\left(n\xi_r < r_{cell} < (n+1)\xi_r\right)$ and $\left(m\xi_\phi < \phi_{cell} < (m+1)\xi_\phi\right)$ for some n and m.

The surface hypothesis calculations then proceed as in equation (5.2) above.

The role that ξ, or precision, plays in the calculation of probabilities is important. At some point, with the exception of possible outliers, the measurement distribution lies completely bounded by the precision terms. The result of this is that any variations in the measurement due to noise are undetectable, and the sensor appears error-free. This is the case with the Milford Sonar. (See Appendix G.) In the course of 500 measurements, not once did the measured range vary. We know that, when using the same transducer but much more precise timers, the sonar error is measurable, and is generally Gaussian in distribution. (See Appendix A.) But when precision is lowered to 1.0 cm., there is no detectable measurement noise.

Equation (5.16) contains two models: one of error, where the distribution is Gaussian, and one of precision, where we have no information of concerning the distribution between the bounds of precision. So there will be cases where the evidence grid should accommodate a bounded-but-unknown (distribution) sensor model.

The second step of the evidence grid methodology is to compute the freespace hypothesis. All evidence grid publications we have reviewed make the same assumptions: the measurement error is the same as the sensor coverage. Note equation (5.14), where the standard deviations of the measurement errors are used to compute the probability of detection, i.e., for each cell in the grid, what is the probability that, if it were occupied, this would have been detected. This coverage analysis is then used to compute the free-space hypothesis, or the region between the measured location and the sensor where a cell, if occupied, would have returned an echo.

We reject this assumption for the following reason. First, as discussed in Appendix B and in Chapter 6, and demonstrated in the proof of concept, the vector sensor has a very narrow azimuth error. Nevertheless, the region ahead of the surface hypothesis where an object would have been detected if it existed (i.e., the freespace) is as wide as the combined beam widths of the two component sonars. More generally, assuming that the fusion of information by the virtual sensor at layer n - 1 of data provided by the virtual sensor at layer n results in reduced error, the freespace hypothesis computed at layer $n - 1$ will be smaller too. By the time layer 1 is reached the error may have shrunk so much that no cell remains in the freespace hypothesis. Thus the standard evidence grid algorithm will always estimate freespace pessimistically. This problem is not apparent in a single layer sensor network, and with the sensor hardware commonly used by evidence grid researchers.

So parameter **propagation** from layer to layer is flawed[16] - as described above it results in a progressively diminished free-space hypothesis. **Generation** is unwarranted, because we can **relay** the data from the physical sensor (layer n) up to the layer where the evidence grid is computed, in order to compute the freespace hypothesis. This does make sense if the freespace hypothesis is considered really to be sensor coverage analysis. Coverage is defined by the combined total area under surveillance by the physical sensors, and this information is directly available from layer n.

The final issue is propagation of the error and precision matrices to the next layer above the grid based fusor, i.e., turning the entire evidence grid into a virtual sensor. Propagation of the precision matrix is simple: the precision of the evidence grid is simply the cell size. Propagation of the error matrix, unfortunately, is only possible under very limited circumstances. As mentioned before, one of the problems of the evidence grid is segmentation: we can determine which cells are free and which are occupied, but it is often difficult to tell if the object occupying cell C_i and the object occupying cell C_{i+1} are different objects or parts of the same object. Assuming that there is one object and we can segment the object, and there are sufficient redundant measurements and surrounding freespace around the target to allow a bivariate Gaussian curve to form, then the required

---

[16] The one case where propagation is warranted is when some n - 1 layer fusor artificially masks some portion of the layer n field of view. An example of this is a search radar where returns below a certain elevation are generally ground clutter, and may be removed before further processing.

measurement vector can be computed: the measured x and y coordinates from the mean of the curve, and the error matrix from the width of the curve. Note, however, than a transformation has taken place; the polar coordinates of the ranging sensor input to the evidence grid have been replaced with the Cartesian coordinates of the grid.

In summary, then: we have modified the evidence grid methodology to include precision, correlated measurements, and to distinguish coverage from measurement error. Each one of these modifications make use of only those parameters available in the virtual sensor data vector. We have also indicated the limited circumstances in which propagation can take place.

5.8.2 Kalman Filters and the Virtual Sensor

Note that the virtual sensor data vector will accommodate the Kalman Filter parameters: the measurement vector corresponds to the Kalman Filter measurement vector **z**, and the measurement covariance matrix (Figure 9) corresponds to the Kalman Filter measurement noise covariance matrix **R**. For propagation purposes, the state estimate $\hat{\mathbf{x}}_k$ can be propagated to the next layer fusor, together with the error covariance matrix $\mathbf{P}_k$. Precision can either be a **generated** parameter, or it can be **propagated** if the Kalman Filter is replaced with a mixed Kalman Filter/Set Theoretic Filter.

One final note: the reader may have noticed that the Kalman Filter output is a state estimate vector, while the virtual sensor interface contains a measurement vector. To complete the "wrapping" of the Kalman Filter, one final step has to be taken: the conversion of this state estimate vector to a measurement vector. Equation (5.4) provides

the solution: we simply treat the measurement vector as a perfect measurement of the state estimate, or $\mathbf{z}_k = \mathbf{H}_k \hat{\mathbf{x}}_k$.

5.8.3 Set Theoretic Filters

The fusion algorithms to this point have assumed either an unknown measurement error or a measurement error with a known distribution. Another possibility remains, however: a measurement error with an unknown distribution that, nevertheless, lies within known bounds. Set theoretic filters for bounded-but-unknown errors are relatively rare in the literature – we have uncovered them only in [98], Hong [49], and Saleem [94], and in recent work by Hanebeck (e.g. [39][40][41]]). For our purposes they do present an effective approach where the error terms are dominated by the precision terms, i.e., where error is undetectable due to the imprecision of the sensor. Two examples of this are provided in the appendices: the range error for the Milford sonar and the range and azimuth errors for the laser range finder.

The measurement precision model for this filter is similar to the Kalman filter error model:[17]

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \tag{5.17}$$

The difference lies in the precision vector **v**, which is modeled as a white process with known bounds but unknown distribution. The objective of the set theoretic filter will be

---

[17] As before, for simplicity sake, only discrete time models and filters are described. For continuous time versions consult [98] for example.

to derive a state estimate $\hat{\mathbf{x}}_k$ from a set of measurement vectors **Z** and, for the purpose of propagation, a set of precision bounds for the fused estimate based on the precision of the measurements.

The set theoretic filter is most easily understood in the case of scalar measurements. Supposed that there are two sensors: S1 and S2, with measurement equations $z1 = Hx + v1$ and $z2 = Hx + v2$ respectively. Assuming that the precision for $z1$ and $z2$ is $\xi 1$ and $\xi 2$, respectively, we define $v1$ and $v2$ by:

$$\begin{aligned} v1 &\in \Omega 1 = \left\{x : x^2 \leq \xi 1^2\right\} \\ v2 &\in \Omega 2 = \left\{x : x^2 \leq \xi 2^2\right\} \end{aligned} \qquad (5.18)^{18}$$

As before, let $Z1 = \{z1_1, z1_2, \ldots z1_n\}$ and $Z2 = \{z2_1, z2_2, \ldots z2_n\}$. If measurements made at the same time so that there are no target dynamics, and if the sensors are registered, then the means of the measurements are the same and either $Z1 \subseteq Z2$ or $Z2 \subseteq Z1$. This is also the case if the measurements are not co-temporal but the target is unchanging. If there are target dynamics, then it is possible that $Z1 \cap Z2 = \varnothing$, and an extrapolation step such as used by the Kalman filter should be used to compute the a priori state estimate (see Section 5.4).

Given $z1$ and $z2$, let

$$\begin{aligned} \Omega_{z1} &= \left\{x : (z1 - Hx)^2 \leq \xi 1^2\right\} \\ \Omega_{z2} &= \left\{x : (z2 - Hx)^2 \leq \xi 2^2\right\} \end{aligned} \qquad (5.19)$$

---

[18] From this point, for simplification, we drop the step index k.

Assuming that $\Omega_{z1} \cap \Omega_{z2} \neq \varnothing$, we can compute the set $\Omega_{est} = \Omega_{z1} \cap \Omega_{z2}$, and $\hat{x}$ as the center of Ω_{est}.[98].

With a little elementary algebra, equation (5.19) can be expressed as

$$\Omega_n = \left\{ x : (z - Hx)\xi_n^{-2}(z - Hx) \leq 1 \right\} \tag{5.20}$$

The vector form then becomes

$$\boldsymbol{\Omega}_n = \left\{ \mathbf{x} : (\mathbf{z} - \mathbf{Hx})^T \xi_n^{-1} (\mathbf{z} - \mathbf{Hx}) \leq 1 \right\}$$

where ξ is the positive-definite precision matrix from the virtual sensor data vector, and n is an arbitrary sensor number. That $\boldsymbol{\Omega}_n$ is an elliptical set can be seen from the following:

$$\text{let } \mathbf{z} = \mathbf{x}, \mathbf{Hx} = \mathbf{m}$$

$$\mathbf{x} - \mathbf{m} = \begin{bmatrix} x-h \\ y-k \end{bmatrix}, \xi = \begin{bmatrix} a^2 & 0 \\ 0 & b^2 \end{bmatrix}$$

$$(\mathbf{x} - \mathbf{m})^T \xi^{-1} (\mathbf{x} - \mathbf{m}) = \begin{bmatrix} x-h \\ y-k \end{bmatrix}^T \begin{bmatrix} a^2 & 0 \\ 0 & b^2 \end{bmatrix}^{-1} \begin{bmatrix} x-h \\ y-k \end{bmatrix}$$

$$= \frac{(x-h)^2}{a^2} + \frac{(y-k)^2}{b^2}$$

In the simplest case, the off-diagonal elements of ξ will be zero, and the axes of the resulting precision ellipse will be parallel to the x and y axes. We have to allow, however, for when this is not the case, such as when the measurement axes do not align with the x and y axis. This is the reason for the off-diagonal terms, as the eigenvalues of ξ, rather than the square root of the matrix trace define the lengths of the semi-major axes of $\boldsymbol{\Omega}$, and define the rotation of the ellipsoid relative to the x axis and y axis.[98]

Returning to the set theoretic filter, the matrix form of the estimation equation

$$\mathbf{\Omega}_{est} = \mathbf{\Omega}_1 \cap \mathbf{\Omega}_2$$

where

$$\mathbf{\Omega}_{est} = \left\{ \mathbf{x} : (\mathbf{z}1 - \mathbf{H}\mathbf{x})^T \xi^{-1} (\mathbf{z}1 - \mathbf{H}\mathbf{x}) \leq 1 \text{ and } (\mathbf{z}2 - \mathbf{H}\mathbf{x})^T \xi^{-1} (\mathbf{z}2 - \mathbf{H}\mathbf{x}) \leq 1 \right\}$$

becomes problematic, since the estimate is not guaranteed to be an elliptical set. This is a problem, since to propagate precision in the virtual sensor data vector we need to define it in terms of the precision matrix. One way out of this conundrum is to calculate a bounding ellipsoid such that

$$\mathbf{\Omega}_{bound} \supseteq \mathbf{\Omega}_{est}$$

A linear feedback estimation model is provided in [98] which is similar to the Bayesian model used by the Kalman Filter. Of interest here is the calculation of ξ, which will be used to propagate the precision matrix to the next layer fusor. There are, of course, an infinite number of bounding ellipsoids. One simple technique is the following:

$$\mathbf{\Omega}_{bound} = \left(\omega \xi 1^{-1} + (1 - \omega) \xi 2^{-1} \right)^{-1}$$

where $0 \leq \omega \leq 1$ and ω is selected "... to minimize some measure such as squared error, determinant, spectral norm, etc..."[98] This equation assumes that the ellipses have coincident centers, i.e. that the sensors are registered. Figure 12 is an example where $\omega = 0.5$; the bounding ellipse is the dotted line. The equation itself should look familiar, since, with the exception of the weighting parameter ω, this is the Kalman Filter error covari-

ance matrix update equation.[19] Other authors describe various other means for obtaining bounding ellipsoids, especially for cases of ellipsoids with non-coincident means. [49] [94] For an algorithm that minimizes the volume of the bounding ellipse see [95].

In summary, then, set theoretic fusion algorithms are supported by the precision matrix of the virtual sensor data vector, and, in turn, provide a means for propagating precision between layers in the fusion hierarchy.

5.8.4 Set-Theoretic Fusion – Final Notes

The similarity between precision and error can be seen by comparing the following equations. Precision defines an ellipse about the measurement vector

$$\Omega_\xi = \left\{ \mathbf{x} : \mathbf{x}^T \xi^{-1} \mathbf{x} \leq 1 \right\}$$

while the following defines the error ellipse contour at k standard deviations:

$$\Omega_\epsilon = \left\{ \mathbf{x} : \mathbf{x}^T \mathbf{R}^{-1} \mathbf{x} \leq k \right\}$$

This leads to a possible heuristic for algorithm determination in sensor management. For example, where the determinant of ξ is small in proportion to $\mathbf{R}$, then choose a statistical estimator. On the other hand, if the determinant of $\mathbf{R}$ is small in proportion to ξ, then

---

[19] This formulation (and quotation) appears in an unpublished paper [13] by Jeffrey K. Uhlmann on the relationship between Covariance Intersection and the Kalman Filter. While the intended use is to bound the intersection of the k-sigma error ellipses, not bounded error or precision, the geometry is the same.

choose a set-theoretic estimator. This, in fact, is effectively what the combined set-theoretic and Kalman Filter described in [39][40][41] does.

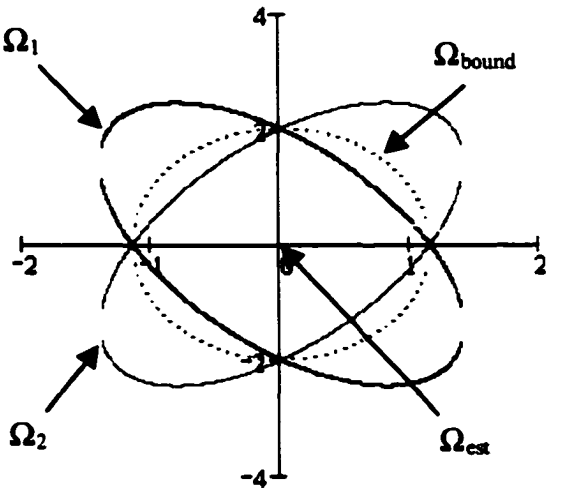

Figure 12: Precision Ellipses

5.9 Correlation

One important intuition concerning the combining of evidence is that information from unrelated sources should be given greater confidence (all other things being equal) than if the sources are related. A further refinement of this intuition is that the measure of confidence should be proportional to the measure of un-relatedness of the sources of information. (Independent confirmation.) From an epistemological standpoint it is relevant to ask whether this relation is causal. This is also possibly relevant in the case of sensor integration, where it is useful in target recognition to ask whether one attribute is causally or accidentally related to another. By the definition of sensor fusion used here

(See Taxonomy), however, causality is immaterial. Measurements that are causally related but otherwise show no statistical correlation do not affect the accuracy of our algorithms; measurements that are causally unrelated but statistically correlated may.

The standard Kalman Filter is a good example of a fusion algorithm that assumes correlation is known.[20] First, in reference to measurement noise vector (**v**) and process noise vector[21] (**w**), the cross-covariances of each (denoted **R** and **Q** respectively) are known. Second, the standard Kalman Filter assumes that measurement noise and process noise are not linearly correlated. Thirdly, it is assumed that the processes $\mathbf{v}_i$ and $\mathbf{w}_j$ are "white," i.e., that for any time index i and for any time index j such that $i \neq j$, $\mathbf{w}_i$ and $\mathbf{w}_j$ are uncorrelated, and $\mathbf{v}_i$ and $\mathbf{v}_j$ are uncorrelated.[3] This is a useful illustration of the three general types of correlation that can occur in sensor fusion

1. Pair-wise correlation between the elements of the measurement vector
2. Correlation between measurement noise and process noise
3. Autocorrelation between noise measured at different intervals

As far as the virtual sensor is concerned, the Kalman Filter makes propagation of (1) easy. We simply propagate the error covariance matrix (**P**) of the filter as if it were

---

[20] "Standard" is used here due to the fact that the Kalman Filter may be modified to account for unknown correlations, for non-white processes, etc.

[21] **Process noise** reflects the uncertainty is the dynamics of the target, as opposed to **measurement noise** which reflects the uncertainty in the sensor measurement.

the measurement noise covariance matrix of a physical sensor, and let this matrix convey the correlation (actually, covariance) for the measurement vector.

As far as evidence grid algorithms are concerned, we have modified (wrapped) those that incorporate a bivariate Gaussian model to accept the covariance of measurement parameters. Those authors who use other models (e.g. [51]) or non-Bayesian algorithms (e.g. [81]) either explicitly or implicitly make an independence assumption, and thus the measurement covariance matrix propagated a lower level virtual sensor is discarded. For all fusion algorithms that do not propagate a measurement error covariance matrix (which is the vast majority), the virtual sensor "wrapper" for these algorithms must generate this matrix. This generation is accomplished either a priori (based on past data of the virtual sensor's performance) or dynamically, by deriving the correlation from the measurement data itself by means of a linear or possibly a rank correlation algorithm. (See [87].)

5.10 The Complete Measurement Vector Specification

To complete this section, we produce in Figure 13 the measurement vector specification for the ranging sensors used in the proof of concept. It should be noted that precision is represented as the trace of the precision matrix, with the off-diagonal terms being zero in the layer n virtual (i.e., the physical) sensor.

$$\begin{bmatrix} x \\ y \\ \theta \\ r \\ \phi \\ \begin{bmatrix} \xi_r^2 & \xi_{ro} \\ \xi_{ro} & \xi_o^2 \end{bmatrix} \\ \begin{bmatrix} \sigma_{\varepsilon_r}^2 & K_{\varepsilon_r\varepsilon_\phi} \\ K_{\varepsilon_r\varepsilon_\phi} & \sigma_{\varepsilon_\phi}^2 \end{bmatrix} \end{bmatrix}$$

x coordinate

y coordinate

sensor pose

measured range

measured azimuth

precision matrix

measurement error covariance matrix

Figure 13: Virtual Ranging Sensor Data Vector

5.11 The Virtual Feature/Symbol Sensor

As promised above, we now show how the data vector for the pixel/signal level virtual sensor can be expanded to feature and symbol level fusion.

The data vector for the virtual feature sensor is more complex than that defined for the pixel/signal level sensor. In addition to inheriting the base sensor attributes (pose, position, etc.), feature measurement data must be included. Depending on the feature, the elements of the measurement vector may include positional coordinates and orientation. The virtual feature sensor implemented in Appendix B shows how these parameters and their associated error/precision metrics can be derived from pixel/signal level data. The unique subvector of the data vector for the feature sensor is a member of an enumerated

set identifying the feature, and some metric representing the confidence in the identification. In the example in Appendix B the identification is fixed. An example of such an enumerated set for the feature sensor described in [10] would be: F = {wall, concave_corner, convex_corner}.

A significant problem results from the range of confidence metrics produced by the commonly used feature level fusion algorithms. Bayesian fusion produces a probability, Fuzzy Logic produces real-valued membership function, Dempster-Shafer produces belief functions,[22] etc. The method we propose to address the propagation of these values is a modification to the wrapper rather than the virtualization layer via polymorphism. This way the fusor is cognizant of the data type for the confidence metric, and can transform/rescale this metric as necessary. This requires that each virtual sensor have an input defined for each type of confidence metric: an input for a probability, an input for a membership function, an input for a mass function, and so on.

[22] For an explanation of Dempster-Shafer evidential reasoning, see Appendix E.

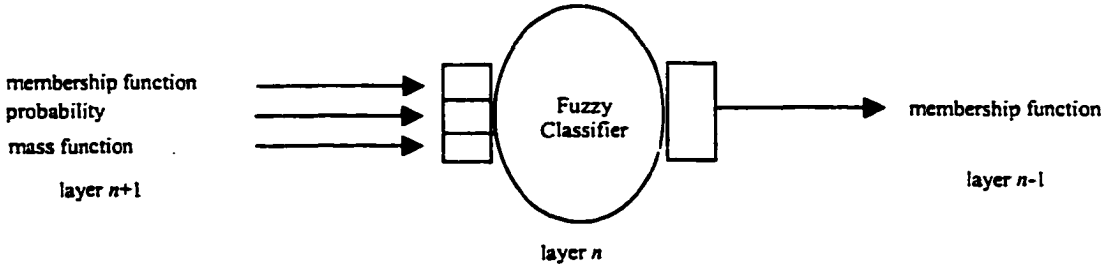

Figure 14: Example - Feature Level Wrapping

In Figure 14, we have represented a single node from Figure 1. Suppose that the virtual sensor supplying data to this node is also a fuzzy classifier. In this case, no transformation is performed at the layer n wrapper where layer $n + 1$ is also a fuzzy classifier. If the virtual sensor feeding this node (i.e., at layer $n + 1$) is a Bayesian classifier, then the wrapper for layer n makes the simple assignment $\mu_f(x_i) = p(x_i \text{ is an } f_n)$, where the left side of the equality is a the fuzzy logic membership function, and the right side is a probability function.[23] Where layer $n + 1$ is a Dempster-Shafer based classifier, then one transformation from belief functions to probability functions that can be used is $Bel(f_n) \rightarrow p(f_n)$ where $f_n \in F$ is a set of features or symbols. Then the layer n wrapper allocates some default feature such that $p(f_{other}) = 1 - \sum p(f_i)$. (See Appendix E for a detailed description of Dempster-Shafer Evidential Reasoning.)

[23] Normalization may be required.

An interesting extension to this virtualization would be to propagate a set of features rather than a single one, each tagged with the computed membership function, mass distribution, or confidence metric. (See Chapter 7.)

5.12 Contact Sensors

One of the most common sensors for inexpensive robots is the contact sensor or bump switch. Basically a binary switch, this sensor triggers upon contact with an obstacle. More often than not these sensors are wired into low-level obstacle avoidance behaviors [54], but there is no reason why they cannot be used for higher level functions, such as mapping.[24]

The complete vector for the contact sensor is shown in Figure 15. The measurement vector for the contact switch is simply the robot pose/position vector displaced by the distance and orientation of the bump switch.[25] The three element precision vector will be based on the precision of the (virtual) sensor used to compute localization, such as a flux-gate compass or odometer. Since this measurement will depend on the accuracy of the robot global localization, the measurement error vector will be derived from the global localization covariance matrix based on the Cartesian x and y coordinates, and

---

[24] In [57] obstacle contact is used for localization.

[25] The robot pose/position may be taken from any point on the robot, such as the center of mass, while the bump sensor may be located at the front of the chassis, or may even form a ring around it, providing contact data in any direction.

sensor pose. There is support in the literature for treating the localization error as Gaussian [35] [15], and so the x and y coordinate error, and the sensor pose will modeled as Gaussian distributions. Thus the data vector provides the necessary ingredients for Kalman filtering, as in the localization algorithms described in [15], [53], and [24]. Simpler filters such as the pseudo-inverse based least squares estimator described in [23] are supported by the measurement vector. Finally, the elements of the covariance matrix can be used to compute the parameters for a Gaussian sensor model.

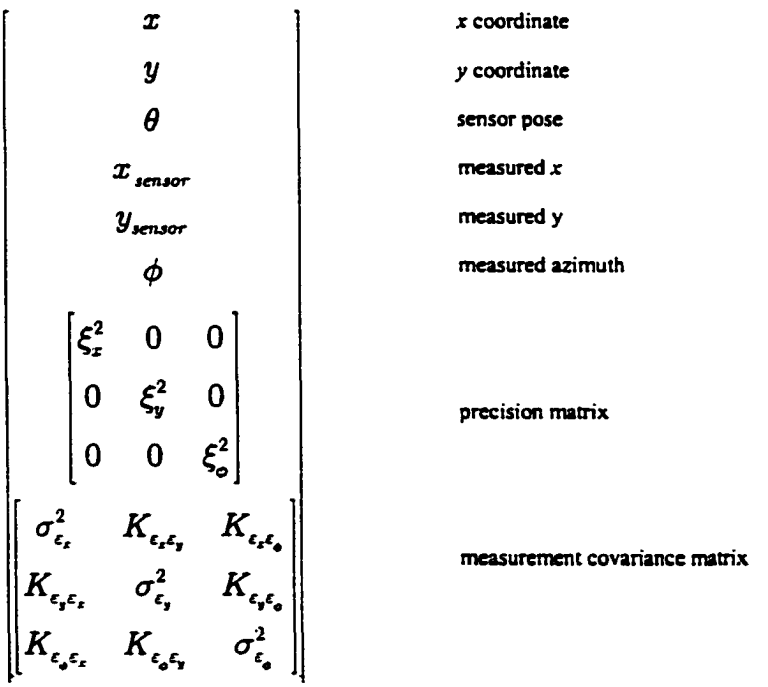

Figure 15: Bump Sensor Data Vector

Chapter 6

Proof of Concept

This section describes an implementation of the concepts described in previous sections. The purpose of this implementation is to prove that the design is feasible, and that the performance penalties exacted by the additional computation is acceptable for a mobile robot.

6.1 Background

The problem domain selected for this proof of concept is single target localization with multiple sensor locations. There are several reasons for this choice. First, it is a domain well suited for a large set of algorithms. Individual measurements can be interpolated or extrapolated, thresholded or smoothed using a wide range of filtering techniques. In addition, localization against a grid is required for the evidence grid methodologies, whether using Bayesian or Dempster-Shafer rules for combining evidence. Second, the fact that evidence grids are not conducive to segmentation (i.e., it is difficult to pick out individual objects by only using the occupancy probabilities) means that it would be difficult to "wrap" the output as a virtual sensor. Finally, the problem space of specular measurements due to multiple targets has been adequately addressed in [62] [76]

[52] and by other researchers. The techniques of surface modeling [62] and pose buckets [76] [52] can easily be accommodated with the virtual sensor architecture.

The next step is to design a sensor architecture for the proof of concept. Based on the characteristics of the sensors available it was natural to combine a pair of sonars into a (virtual) vector sonar, and a sonar and laser range finder into a (virtual) laser-limited sonar. At the top of the pyramid in Figure 16 is the evidence grid algorithm.

Beyond the basic sensor architecture are the algorithm choices. For example, while we could use either a Bayesian or Dempster-Shafer algorithm for combining grid evidence, none of the statistical filters would have been appropriate. In general, though, we could have "plugged" a number of alternative algorithms in at other layers and achieved a viable system.

For the raw sonar data, due to the presence of outliers, we implemented a simple gating function. We chose a simple thresholding function of 4σ based on observation of typical measurement errors and in order not to "clip" the Gaussian tails of the measurement error.[26]

---

[26] [24] has an excellent discussion of using gating functions for pre-processing prior to using an Extended Kalman Filter. While he prefers a sigmoid weighting function to the step function described above (see p. 189), the step function is acceptable given the

Following the pre-processing, sets of measurements from each of the paired sonar measurements are sequentially smoothed by use of a simple Kalman filter to produce a single range estimate from each. At this step either a second Kalman filter could be used to batch process the paired estimates, or a simple trigonometric identity to extract a more precise azimuth measurement (see Appendix B). The latter was chosen. Note that this architectural choice also resulted in a non-zero covariance of the range and azimuth, since the azimuth is computed from the range estimate.

As for the paired sonar and laser range finder, the sonar measurement was combined without filtering with the laser measurement. The fused data vector used the range and range variance/precision of the sonar, and the azimuth and azimuth error/precision from the laser range finder. Finally, the data vectors from both virtual sensors (vector and laser limited) were combined in the evidence grid. Figure 16 depicts the complete architecture.

---

relatively "clean" problem domain of this proof of concept. Also see [38] for a chi-square gating function.

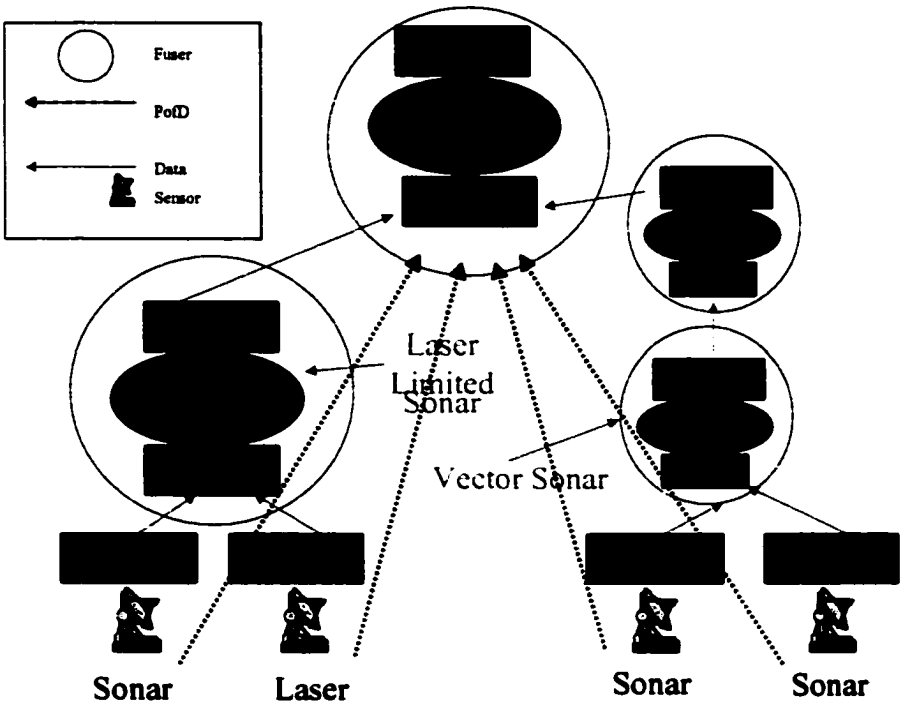

Figure 16: Proof of Concept

6.2 Implementation: Vector Sensor

The details of the implementation of the (virtual) vector sensor are described in Appendix B. The measurement covariance matrix **R** could have been calculated (propagated) from the two input error covariance matrices (one from each component sonar). Because the next step (Kalman Filter) would be sequentially rather than batch processed, it was decided to generate a series of 500 measurements. It was from this series of measurements the error covariance matrix was generated.

6.3 Implementation: Kalman Filter

The output of the vector sensor has a measured range variance of 0.170577 (millimeters), and an azimuth variance of 0.005338 (degrees) over a population of 500 measurements. Since the both the azimuth and range are derived geometrically from the range measurements of the two sonars, one would expect some correlation. Indeed, the covariance of range and azimuth is 0.001693. Thus the measurement covariance matrix for the vector sensor is:

$$\mathbf{R} = \begin{bmatrix} 0.170577 & 0.001693 \\ 0.001693 & 0.005338 \end{bmatrix}$$

Instead of sending an arbitrary measurement vector to the occupancy grid, we demonstrate the properties of the virtualized sensor by passing the 500 measurements through a simple Kalman Filter, and then presenting the smoothed data as the data vector of yet another virtual sensor.

The discrete Kalman Filter Equations are reproduced below.[27]

$$\begin{aligned} \mathbf{K}_k &= \frac{\mathbf{P}_k^- \mathbf{H}_k^T}{\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k} \\ \hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-) \\ \mathbf{P}_k &= (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}_k^- \end{aligned} \tag{6.1}$$

[27] See Chapter 5 for a fuller explanation of Kalman Filters.

$$\begin{aligned} \hat{\mathbf{x}}_{k+1}^{-} &= \mathbf{\Phi}_k \hat{\mathbf{x}}_k + \mathbf{B}_k \mathbf{u}_k \\ \mathbf{P}_{k+1}^{-} &= \mathbf{\Phi}_k \mathbf{P}_k \mathbf{\Phi}_k^T + \mathbf{Q}_k \end{aligned} \tag{6.2}$$

where equations (6.1) update the state estimated based on measurement data, while equations (6.2) update the state estimate based on system dynamics.

The implemented Kalman Filter is far simpler than equations (6.1) and (6.2). The target is motionless, so the system dynamics in (6.2) are not used. The initial system state estimate $\hat{\mathbf{x}}_0$ is set to $\mathbf{z}_0$, i.e., the initial estimate is the first measurement vector. $\mathbf{R}_k$ for $0 \leq k \leq 499$ is derived from the virtual vector sensor as described above. Note that while $\mathbf{R}$ remained static through the experiment, the Kalman Filter and the virtual sensor both allow for $\mathbf{R}$ to change between measurements (thus the 'k' subscript), reflecting changes in the underlying physical sensors.

$\mathbf{P}_0$, as is often the case, is set to $\mathbf{R}_0$, indicating that the initial error covariance is based purely on measurement noise. If there were state dynamics then equations (6.2) would add process noise as the filter evolves.

Finally $\mathbf{H}_k$, the measurement matrix, is set to the identity matrix, since all measurements of state variables are direct, and both range and azimuth are measured. As described above in the section on the virtual vector sonar, the Kalman Filter could have been implemented one layer down inside the vector sonar. In this case $\mathbf{H}$ would relate the two range measurements of the two sonars that are part of the vector sonar to the single range and azimuth elements of the vector sonar data vector.

Figure 17 and Figure 18 show the smoothed range and azimuth estimates as the filter evolves.

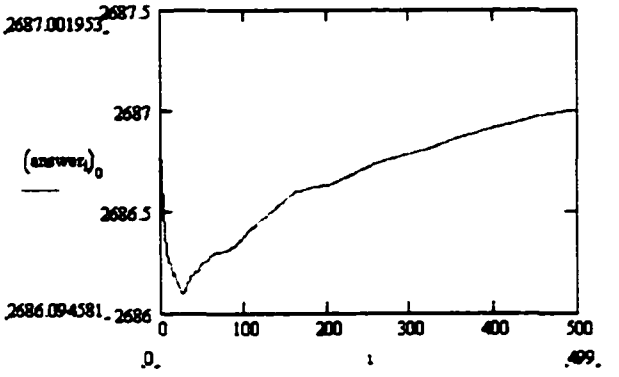

Figure 17: Smoothed Range Data

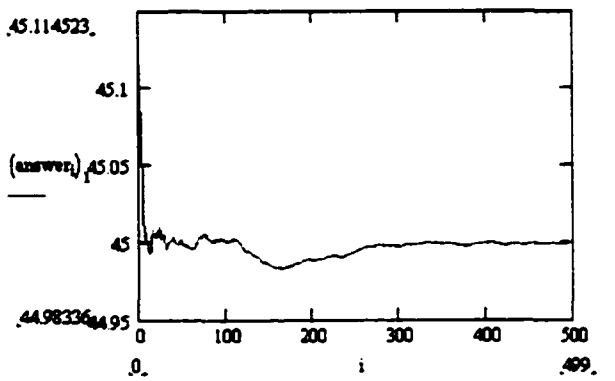

Figure 18: Smoothed Azimuth Data

Because the Kalman Filter maintains the second moments in **P**, we pass

$$\mathbf{P}(499) = \begin{bmatrix} 0.00034 & 0.000003 \\ 0.000003 & 0.000011 \end{bmatrix}$$

as the measurement covariance matrix for this virtual sensor.

6.4 Implementation: Evidence Grid

Chapter 5 contains a detailed discussion of evidence grids in general. We review the salient points for this proof of concept here.

The basic algorithm used by nearly all researchers has two parts. The first part updates the confidence that a cell is occupied (surface hypothesis) given new evidence (i.e., a new measurement) by combining prior confidence with a weighted measure of the new evidence. Equation (6.3) provides the Bayesian rule of combination used in the proof of concept.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{6.3}$$

The second part to the algorithm is the free-space computation, which computes, for each cell visible to the sensor and closer to the sensor than the first detected object, the confidence level that the cell is unoccupied. This normally takes some form of equation (6.3) above, although the exact implementation details vary from researcher to researcher. We chose to follow [29] rather than [62] in this implementation based on the weakness of the latter's approach to modeling environmental factors as described above. (See Section 5.3.2.1.) In addition, no attempt was made to implement any of the data dependency approaches of [6], [7], [51], [52], or [76]. The former approach (region growing) can be easily implemented on top of the actual data fusion, while the latter approaches ("pose buckets") are better characterized as sensor management than sensor fusion.

The sensor model is at the core of this algorithm. The purpose of this model is to provide the left part of the numerator and the denominator for equation (6.3). The choice was made to remain with the bivariate distribution used by many of the occupancy grid researchers (which accommodates the bounded-but-unknown case as described in Section 5.8), although some other distribution can be allowed for by using the communicating mathematical objects method described in Section 7.1. Because of the correlated measurement vector of the vector sonar, however, we have been forced to go beyond the other grid implementations to include the correlation term in the bivariate normal distribution. This is expressed in equation (6.4).

$$f(r,\theta)=\frac{1}{2\pi\sigma_r\sigma_\theta\sqrt{1-\rho^2}}\exp\left(\frac{1}{2(1-\rho^2)}\left(\left[\frac{-(r-D)^2}{\sigma_r^2}\right]+\left[\frac{-(\theta)^2}{\sigma_\theta^2}\right]-2\rho\left[\frac{r-D}{\sigma_r}\right]\left[\frac{\theta}{\sigma_\theta}\right]\right)\right) \quad (6.4)$$

In equation (6.4) the term $(r - D)$ describes the difference between the measured range r and the range to any particular cell D. θ is equal to the difference between the target azimuth (i.e., measurement azimuth + sensor pose) and the cell azimuth. These parameters are easily extracted from the virtual sensor measurement vector. $-1 \leq \rho \leq 1$ is the correlation coefficient of the range and azimuth measurements, and is computed from the covariance matrix in the measurement vector as follows:

$$\rho=\frac{K_{ro}}{\sigma_r\sigma_o}$$

where the denominator is derived from the measurement $\mathbf{tr}(\mathbf{R}_k)$.

Finally, the standard deviations of the range and azimuth are simply the square roots of the diagonal terms in the measurement vector covariance matrix.

The next step brings a thorny implementation issue. It will be recalled from Section 5.8 that probabilities are derived by computing the double integral of the distribution bounded by the measurement precision (both range and azimuth). The computational complexity of any common numerical quadrature is further compounded by the $O(mn)$ iterations for a gird of size m by n cells. This performance bound can be reduced somewhat by using [9], but the penalty is probably still unacceptable (if one requires real-time performance) for sizeable grids.

The method employed in this proof of concept is a familiar one from pre-MATLAB days: table lookup. The technique is to pre-compute the distribution off-line, read it into a suitable data structure during program initialization, and perform a linear-time lookup for the probabilities. The detailed steps are as follows. First, a cumulative distribution function for the standard bivariate normal distribution is computed. The standard distribution is one for which both standard deviations are 1, and the mean of the distribution is 0. We wrote a short program using the National Institute of Standards and Technology Dataplot software to generate the table between the bounds $-4.0 \le x, y \le 4.0$ and $-1.0 \le \rho \le 1.0$ with single decimal place accuracy for a table of size 81×81×19×sizeof(float) = 997,272 bytes. The equations for normalizing non-standard standard deviations are provided in equation (6.5).

$$x = \frac{(D-r)}{\sigma_r}$$
$$y = \frac{\theta}{\sigma_r} \tag{6.5}$$

Finally, the probability can be computed from the cumulative distribution as follows.

$$\int_a^b \int_c^d f(x,y)\,dydx = cum(b,d) - cum(b,c) - cum(a,d) + cum(a,c) \tag{6.6}$$

where $cum(x,y) = \int_{-\infty}^{x} \int_{-\infty}^{y} f(x,y)\,dydx$.

This equation subtracts the cumulative distributions of the lower limits of integration from the higher limits, and adds back the overlapping area. This equation is easily derived by repeated changes to the limits.[28]

Some notes on the methodology: this algorithm is a classic tradeoff between time complexity and space complexity, with accuracy as the losing party. Expanding the

---

[28] Equations (6.5) and (6.6) are simple bivariate extensions of univariate forms common in many statistics texts. See, for example, [QUIRIN78] pages 267, 270 or [14] pages 112-113.

lookup table another decimal place for all three parameters (both standard deviations and correlation) results in a table of size 801×801×181×sizeof(float) = 464,519,124 bytes, which is certainly at the fringe of what is acceptable for physical RAM. This can be easily ameliorated, however. Changing from floating point to fixed point quickly halves that number. The table size can be reduced even further by employing symmetries in the probability distribution itself. Finally, the structure and use of the table make it admirably suited for virtual memory. Each ρ can be allocated a page, and the appropriate ρ can be paged in at run time as needed. Since ρ does not change during computation, only, at worst, between measurements, page faults should not be a significant performance bottleneck.

The alternative is to compute the probabilities during execution. Since quadrature is clearly too slow for real-time computation, the alternative is to use an approximation such as [9]. AS R80, and the called function AS 66 are fairly fast: on a 300mhz Pentium II each cell probability takes approximately 0.0043 seconds to compute. Even this performance is probably unacceptable, since (worst case) a 40×40 grid will take 6.88 seconds to recompute, which is far higher than the revisit rate of a single sonar, let alone that of a laser range finder. This performance can be improved by hardware changes; since the cell probabilities are computed independently, the algorithm is easily parallelizable.

6.5 Results

The AOI was an area 4.0 by 4.0 meters square, tessellated into 1600 16 by 16 10cm cells. A target was placed in cell [19,19]. The vector sonar was positioned in cell

[0,0] (Location 2), while the laser-limited sonar was placed in cell [19,0] (Location 1). Thus the pose/position vectors for each sensor and the positional vector for the target was:

$$\text{Vector} = \begin{bmatrix} 0 \\ 0 \\ 45.0 \end{bmatrix} \quad \text{Laser} = \begin{bmatrix} 1900 \\ 0 \\ 0.0 \end{bmatrix} \quad \text{Target} = \begin{bmatrix} 1900 \\ 1900 \end{bmatrix}$$

The AOI is depicted in Figure 19.

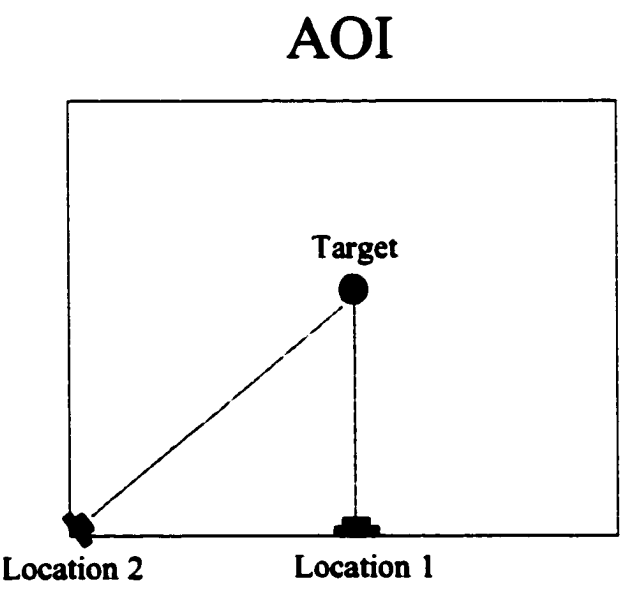

Figure 19: Area of Interest

Each sensor was calibrated as described in Appendices A-C. Then a series of 500 measurements was made for each component of the vector sonar (to support the Kalman Filter), 500 measurements were made for the sonar associated with the laser and gated as

described above. A single range measurement was made for the laser. To illustrate the effect of the successive fusion operations, the results at each step were updated in occupancy grids. These are reproduced below in Figure 20 through Figure 21.

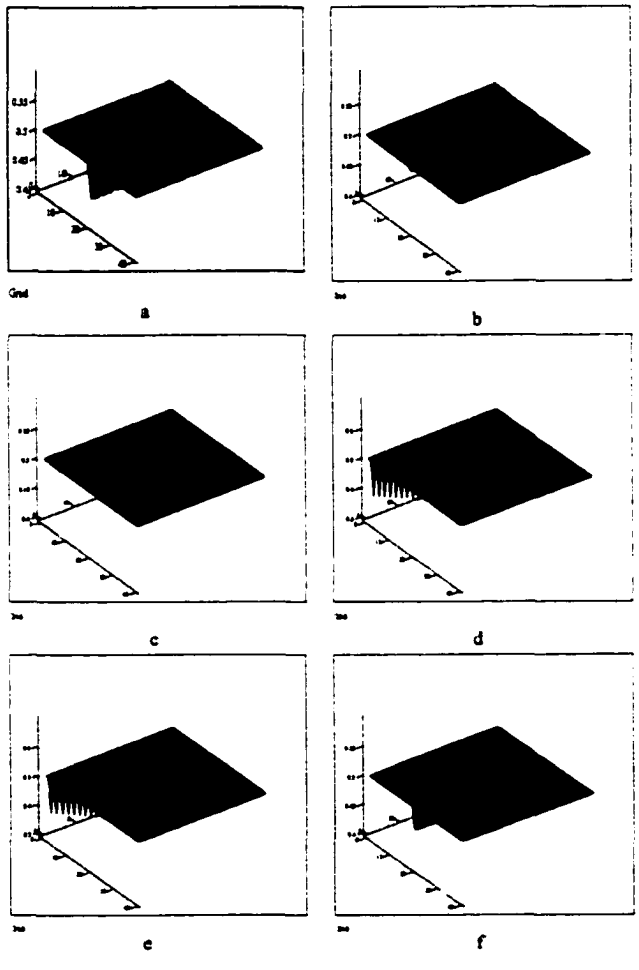

Figure 20: Results (1 of 2)

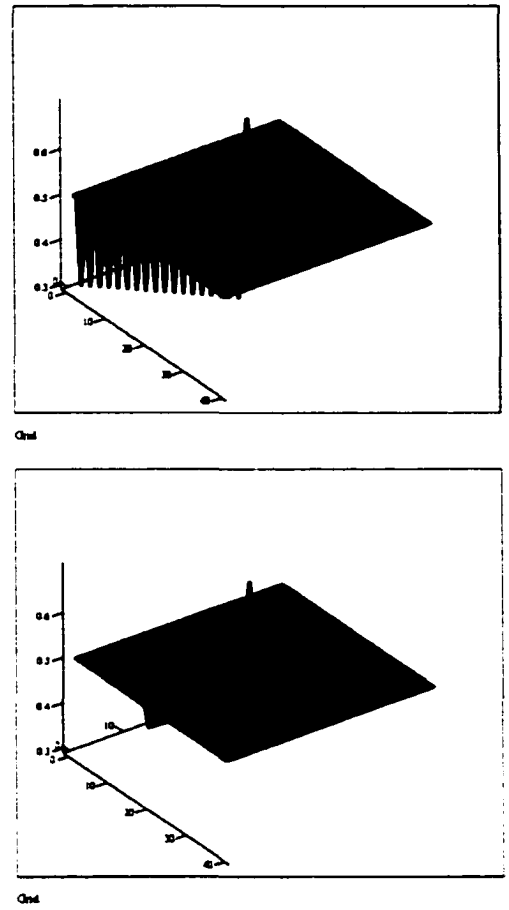

Figure 21: Results (2 of 2)

In each figure cell $C_{0,0}$ is located in the middle left-hand side. The probability of occupation has been normalized to 0.5 (non-informative prior), which is represented by the y axis. The x and z axes are labeled by cell number.

Figure 20.a depicts the laser range finder. Both the range and azimuth errors are below the level of precision, so the freespace hypothesis (the valley between the target and the sensor) and the surface hypothesis (the hill at the target location) represent sensor precision only. Figure 20.b depicts the sonar coaxial with the laser, and, like Figure 20.c

which represents one of the two vector sonar's sonars, shows the distinct cone of the sonar azimuth error. Figure 20.d shows the vector sonar data after Kalman Filtering. Both Figure 20.d and the top graph in Figure 21 show the results of fusion using the standard algorithms of Elfes and Konnolige.[29] Note the incorrectly narrow free-space hypothesis in each. Finally, the bottom graph in Figure 21 depicts the results using the improved evidence grid algorithm.

6.6 Conclusions

The proof of concept demonstrated the objectives (and limitations) stated in Chapter 1:

- Sensor Virtualization: A single abstraction was devised for a group of heterogeneous, but commensurate sensors. This abstraction was shown to propagate, within the restrictions noted in Chapter 5, to layers above that of the physical sensor.

- Algorithm Neutrality: Within the limitations noted in Chapter 5, the choice of wrapped fusion algorithm was independent of the sensor type, and vice versa.

- Robustness: This was demonstrated in two ways. First, by showing that any stage could be "plugged into" the top-layer algorithm (evidence grid), thus simu-

[29] See Section 5.8.1 for an explanation of coverage analysis.

lating the complete failure of a lower-layer sensor. Second, by maintaining a measurement covariance matrix within the virtual sensor and propagating it to the next layer fusor, we have demonstrated that the architecture can accommodate degradation of accuracy in the measurement parameters.

- Acceptable Performance: The only additional computation is involved in wrapping the evidence grid algorithm, and we have demonstrated how this can be reduced to equivalence with methodologies that do not perform the iterated integral.

- Decentralization: The data vector has been shown to be necessary and sufficient for a varied set of fusion algorithms, which need not be co-located with the sensor.

Chapter 7

Future Directions

It became apparent in the course of the research described in this dissertation just how fruitful a field this was for exploration. There was a continual process of scope delineation to meet time and space constraints; in this section we remove the constraints and explore topics that we (and others) will be addressing in the future.

7.1 Communicating Mathematical Objects

The virtual sensor data vector described in Chapter 5 communicates error and precision based on a mathematical description of sensor performance. The architecture we have described requires that this model be maintained by the virtual sensor, i.e., the sensor model for layer n is maintained at layer n, not in layer $n-1$ (or layer 1 for that matter). In some cases, due to processor limitations, we chose to simulate this, and maintain the sensor model at layer 1.

While this approach is ideal for modularity purposes, it does have several drawbacks.

- Footprint: computation of the sensor model on the same hardware platform as the virtual sensor may exact a stiff computational requirement. This is especially

problematic if this platform is a mobile robot or a remote physical sensor, which often have stringent physical limitations such as cost, size, weight, power consumption, or heat dissipation. Performing the computations locally and, at the same time, providing acceptable performance may involve unacceptable hardware requirements. Thus it is advisable to limit the computational requirements for the remote processor as much as possible.

- Implicit Error Distribution: Layer $n-1$ may require more explicit knowledge of the sensor model than simply error and precision. For example, optimal choice of a fusion algorithm at layer $n-1$ demands information on the error distribution used at layer n.

- Bandwidth/Latency: In a worst case analysis, for each measurement the evidence grid would need to interrogate the virtual sensor $n \times m$ times for an $n \times m$ grid in order to compute freespace and surface hypothesis. This clearly exacts a performance penalty on the sensor network.

One solution to this problem is to pre-define a set of models within each fusor. When a virtual sensor connects to the fusor, the sensor type is communicated in the initial handshaking, and the fusor selects the appropriate model. The problem with this solution is that all possible sensor types must be known in advance to the fusor. While it is conceivable that all possible **physical** sensor models can be determined *a priori*, it is inconceivable that models for all possible **virtual** sensors can be known in advance.

A better solution to the problem is to communicate the layer n sensor model to layer $n-1$ as part of the initial handshaking. We initially investigated mobile code as a possible mechanism to communicate this model, but quickly rejected it for several reasons. First, mobile code exacts its own penalties in terms of bandwidth and performance. Second, we want to communicate how to compute the sensor model, and not necessarily the code to compute it. It should be the fusor that determines which numerical algorithm (and associated parameters) to use to compute the model based on local performance and accuracy requirements.

An intriguing possibility came to mind while reading [66]. Le describes the concept of a computational server, where equations are submitted by a remote client formatted in content MathML, computed using a suitable computational tool (Maple in this case), and the results are returned on an XHTML formatted in presentation MathML.[30] This leads quite naturally to the notion of communicating the sensor model using content MathML during initial handshaking between fusor and virtual sensor. Figure 22 shows a simple example using a normal probability distribution for the sensor model. Note that

---

[30] Math Markup Language, or MathML, is an XML markup language designed to capture a rich subset (and extensible) of mathematical notation. It comes in two flavors: **presentation**, which defines how the notation is intended to be displayed, and **content**, which provides the semantics for the notation. The MathML 2.0 proposed specification is available at http://www.w3c.org/

both the model and model parameters can be communicated in the same MathML document.[31]

Use of an XML language presents some intriguing possibilities. For communications between virtual sensor and fusor the simple object access protocol (SOAP) provides a simple mechanism for transmitting serialized MathML documents. Universal Description, Discovery, and Integration (UDDI) provides a method for virtual sensors to advertise, and for fusors to request information services. The document object model (DOM) and the simple API for XML (SAX) provide alternative methods for parsing MathML documents. XML Query supports queries on parsed MathML documents.

More than just transmitting the sensor model in MathML, MathML provides an excellent interface description language for the data vector itself. MathML content allows for data typing via type attributes such as:

```
<ci type = "vector"> v </ci>
```

where identifier v is an unspecified element vector, and type declarations such as

```
<declare>
 <ci> A </ci>
  <vector>
   <cn type = "integer"> 1</ci>
   <cn type = "real"> 1.33</ci>
   <cn type = "real"> 2.0</ci>
  </vector>
</declare>
```

---

[31] In XML terms, an segment of XML (or MathML) text is referred to as a "document."

where A is declared as a three-element vector consisting of one integer and two real typed elements. This typing allows for a polymorphic interface to the algorithm wrapper. (e.g. Figure 14) We are currently drafting a publication for submission to the **International Journal for Information Fusion** exploring these concepts.

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
<!--Range Error Variance-->
<mrow>
 <apply><eq/>
  <cn type = "constant">1.5</cn>
  <ci type = "real">var_range</ci>
 </apply>
</mrow>
<!--Sensor Model-->
<mrow>
 <apply><eq/>
   <apply><times/>
    <apply><divide/>
     <cn type = "constant">1.0</cn>
     <apply><root/>
      <apply><times/>
       <apply><root/>
        <ci type = "real">var_range</ci>
       </apply>
       <apply><times/>
        <cn type = "constant">2.0</cn>
        <cn type = "constant">&pi;</ci>
       </apply>
       <cn type = "constant">2.0</cn>
      </apply>
     </apply>
    </apply>
   </apply>
   <apply><exp/>
    <apply><divide/>
     <apply><minus/>
      <apply><power/>
       <apply><minus/>
         <ci type = "real">D</ci>
         <ci type = "real">r</ci>
       </apply>
        <cn type = "constant">2.0</cn>
      </apply>
     </apply>
     <apply><times/>
      <cn type = "constant">2.0</cn>
      <ci type = "real">var_range</ci>
     </apply>
    </apply>
   </apply>
  <ci type = "real">probability</ci>
 </apply>
</mrow>
</math>
```

Figure 22: MathML 2.0 Sensor Model

7.2 Information Fusion

The US Air Force Scientific Advisory Board (SAB) in studies conducted in 1998 and 1999, define an information based architecture for conducting net-centric warfare called the Joint Battlespace Infosphere (JBI).[92] The basic tenets are fairly straightforward. An information producer (which may be a sensor, a human analyst, a predictive simulation, etc.) advertises its services to a Broker. This producer is responsible for publishing information objects to information consumers, who subscribe their information requests to the Broker. The information objects consist of a manifest, and a payload. The payload is the information itself, while the manifest describes certain properties about the information and information publisher: time/date, information quality, information source, etc. The final piece in the JBI puzzle is the fuselet (a nod to the term applet), which behaves as an information consumer, information transformer, and information publisher publishing transformed information objects.[32]

While our work on sensor virtualization predates, and has had no impact on the SAB study, there are useful similarities. The data vector contains many of the features of

---

[32] An initial experiment was conducted by the Air Force Research Lab (AFRL) and the MITRE Corporation at JEFX00. Work is on-going through seed money at AFRL and several Air Force sponsored research programs at MITRE.

an information object, and the wrapped fusion algorithms may be seen as fuselets.[33] While the architecture we devised is data pull (i.e., the fusor interrogates the virtual sensor), it would not be difficult to incorporate virtual sensors into the JBI publish/subscribe or data push paradigm.

The more difficult problem is to extend the simple sensor error and precision parameters to the information "pedigree" described in the SAB studies. Compared to maintaining error and precision for a physical sensor and propagating these values through virtual sensors, the problem of "quality" for an arbitrary information object becomes quite complex. In fact, the greater the distance from a directly measured property of a physical object, the more "subjective" the property becomes. Issues of data dependency are similarly difficult to objectify. Shafer goes as far as to state:

> Whether two items of evidence are independent in a real problem is a subjective judgement, in the belief function as in the Bayesian approach. There is no objective test. [99]

This statement was echoed by Dr. Subrata Dass of Charles River Analytics, one of the AFRL JBI fuselet contractors, when we questioned him on his Bayesian Net approach to information fusion.[34] We believe that by grounding the information net in physically measured properties as much as possible, and by providing appropriate error, precision, and dependency propagation through the layers of information fusion, assignment of

---

[33] Although there is currently great disagreement among the JBI community on exactly what fuselets are.

[34] AFRL JBI Principle Investigators meeting, Fayetteville New York, May 2001.

subjective values to attributes becomes a last, rather than first resort in the information fusion process.

Related to information fusion are the issues concerning feature level and symbol level fusion as described in Section 5.11. MathML, once again, provides a mechanism by which features and symbols, and their associated confidence metrics, can be communicated between layers. For example, let layer *n* be a symbol layer fusor that identifies objects from the set <jet, helicopter, unknown>. This set is built by using the set container constructor as follows:

```
<declare>
 <ci> Symbol_Set</ci
 <set>
  <ci>jet</ci>
  <ci>helicopter</ci>
  <ci>unknown</ci>
 </set>
</declare>
```

Then probabilities can be assigned by layer n and passed up the chain by a simple assignment:

```
<apply><eq/>
 <ci>jet</ci>
 <cn>0.45</cn>
</apply>
```

7.3 Sensor Management

As stated earlier, sensor management can be seen as the complement to sensor fusion. While sensor fusion asks how we can combine sensor data to improve an estimate, sensor management is the scheduling and tasking of sensor resources to ensure the optimal collection of this data. The process model, as described in [38] is:

> Outputs from sensor management include sensor control data (i.e. commands, thresholds, data collection parameters such as rates, signal-to-noise ratio criteria, etc., and look angles) ... Inputs to sensor management are a priori information from a database such as sensor characteristics (e.g. location, sensor type, model parameters), observation model parameters, system constraints, and environmental parameters.

Hall's inputs are covered, with the exception of the environmental parameters, by the virtual sensor data vector. Hall's outputs are covered by the control vector and measurement vector. As described earlier, more work needs to be done in the area of coverage analysis and its propagation through layers of virtual sensors, and whether the control vector of layer n will be managed by layer n - 1, or by a centralized sensor management function. However this is decided, the uniform abstraction of the virtual sensor interface layer will undoubtedly simplify the task of sensor management.

7.4 Sensor Integration

The focus of this dissertation has been sensor fusion, wherein the data vector emitted from the fusor is the same as, or related by some explicit linear or non-linear transformation, to the data vector input. In short, both input and output data are commensurable. This, by no means, is the only use to which the combination of sensor data is put. We broadly lump all others under the heading of sensor integration, which is to include moving between fusion levels as described in [69] and level 2 (situation assessment) and level 3 (threat assessment) fusion as described in [38]. Generally speaking, the objective of sensor integration is to increase the total information about a target. Unlike sensor fusion, then, redundant information is of little value, since it is assumed that the

commensurate sources have already been fused, and uncertainty about any set of properties has been reduced to some acceptable level.

Target recognition is a perfect example of sensor integration. The objective of the exercise is to identify a target by sensor data. In the broadest terms, a certain set of potential matches are defined: {Jet, Helicopter, Weather Balloon}. A set of distinguishing properties are defined for each, all of which are detectable by or derivable from available sensor data. The algorithm used to derive an identification (and possibly a confidence metric in the identification) can be a Bayesian Net, Neural Net, Fuzzy Logic algorithm, Dempster-Shafer Evidential Reasoning, model based target recognition, and so on.

The challenge to the vector sensor is related to that discussed for information fusion above: How to propagate information quality metrics through the integration process? The algorithms themselves are problematic. Each describes confidence/uncertainty in different terms: as a random variable, mass distribution, set membership function, etc. Algorithm wrapping will require a transformation function be defined to support propagation. Note that, once again, MathML provides some intriguing possibilities. As described in Section 7.2, typing by means of <declare> supports polymorphism via the base types (integer, real, rational, complex-Cartesian, complex-polar, and constant) and the containers constructor (interval, list, matrix, set, vector). A confidence parameter that is a random variable can be defined as the following closed interval:

```
<interval closure = "closed">
 <ci> 0.0 </ci>
 <ci>1.0 </ci>
</interval>
```

Thus MathML and XML in general can be used to tag the confidence "type" as a random variable, mass distribution, fuzzy set, and so on, and the container constructors can be used to pass the corresponding parameters.

Conclusion

Within the scoping of the problem delineated in Chapter 1, this research has achieved many of its objectives. A sensor data vector has been formulated that is sufficient to support a large set of fusion algorithms common to signal and pixel level fusion. In the case of evidence grids, the research has incidentally uncovered significant problems in prior efforts, and offers modified algorithms to avoid these problems. Where necessary, we have also shown how these algorithms can be "wrapped" to present an output vector consistent with the sensor data vector, thus providing for a virtual sensor. This sensor virtualization is essential for hierarchical, distributed, or decentralized fusion architectures. In a proof of concept we implemented a hierarchical distributed architecture, and showed how it could support heterogeneous sensors, heterogeneous algorithms, and dynamic configuration/reconfiguration. We have also described how the constraints imposed in the problem scoping can be relaxed: Sections 5.11 and 7.2 show how the architecture can be extended to feature and symbol level fusion. Our work also proves that the simple exposing of error parameters proposed by Henderson [45][46] is insufficient across the panoply of possible sensor hardware, and that the a priori knowledge of arbitrary virtual sensor models by the fusor is not feasible. Chapter 7 shows how the sensor model can be extended to error models other than Gaussian or bounded-but-unknown by communicating mathematical objects, which also shows promise as an approach to extending the virtual sensor specification to communicating sets of features or symbols as required by feature and symbol level fusion, and sensor integration.

The work presented in this dissertation can be viewed in the context of the larger problem facing Artificial Intelligence in general, and knowledge based systems and data mining in particular. The general problem that must be addressed in any of these fields concerns theories of evidence: how information should be combined to support confidence that a certain assertion is correct. Information Fusion involves the study of algorithms for combining information into a quantitative assessment of this confidence factor. Sensor Fusion is the subset of Information Fusion where the information objects are derived from sensors. The justification for narrowly scoping the Epistemic problem to Sensor Fusion was detailed in Chapter 4: that it contributes to solutions of localization and mapping problems for Mobile Robots. Restricting information fusion to sensor fusion is not possible in symbol and feature level fusion, where properties of features (e.g. walls, corners) and symbols (e.g. humans, aircraft) are either a priori known to the algorithm, or learned by generalization from training examples. Even at the signal level, a priori (i.e., non-sensor) information is combined with sensor data. In the Kalman Filter, for example, the state transition matrix that describes the target dynamics is combined to create the a priori state estimate, and thus an element of information fusion is introduced into the a posteriori state estimate. What enables this information to be fused, however, is that it is **grounded**: a relation is defined between this information and some properties of a real object that are verifiable and measurable. To extend the result of this dissertation to Information Fusion in general will require the specification of properties of information objects that are objective, quantifiable, and commensurate.

Appendix A: Sonar

This appendix describes in greater detail the design, implementation, and testing of the ranging sonar measurement equipment used to collect data for this thesis. First, the geometry of the virtual sensor is described. Then we describe the hardware and software implementations of the sensor. Finally we examine some of the properties of this sensor.

9.1 Sensor Geometry

A sonar is a time-of-flight ranging sensor that functions by timing the round trip of the pulse from the transmitter transducer to a reflecting target and then the returning echo back to a receiving transducer.[35] The target range is determined by

$$r = \frac{ct}{2} \tag{9.1}$$

where t is the time from transmission to reception, and c is the speed of sound in the particular medium. c varies given temperature, humidity, and air pressure. It can be generally calculated by

---

[35] Following [50], the signal from the transducer to the target is termed the **pulse**; the return signal from the target to the receiving transducer we term the **echo**.

$$c = \sqrt{\gamma RT} \tag{9.2}$$

where γ is the ratio of specific heats, R is the gas constant, and T is the ambient temperature in Kelvin. Thus the speed of sound at room temperature in dry air is approximately 344 m/s.

Sonars may have separate transmission from reception transducers or, to save costs and ensure precise registration, the same transducer may serve both purposes. Borrowing terminology from radars, if the transmitter and receiver are separate, then the sonar is **bistatic**; otherwise the sonar is **monostatic**. A bistatic sonar may have multiple receivers, for reasons which will be described below. If this is the case, then the collection of receivers is termed an **array**.

Figure 23 depicts the geometry of a typical sonar. The transducer, in this case monostatic, both transmits the pulse and receives the returning echo. The pulse propagates from the transducer in a conical pattern, the width of the cone (**beam width**) inversely proportional to the diameter of the transducer. If the pulse is blocked by some object lying within this pulse cone, then a portion of the energy is absorbed, and a portion of the energy is reflected. How much energy is absorbed depends on the acoustic properties of the target. The direction that the reflected energy takes depends on the surface texture and angular relation of the target to the pulse. Sufficient energy may be absorbed and/or reflected away from the transducer that the return echo is undetectable. Energy may be reflect off the target away from the transducer, bounce off other objects, and then

return to the transducer. This is known as **specular reflection**. The former case is known as the **specular model**; the latter as the **diffuse model**. [93]][36, 37]

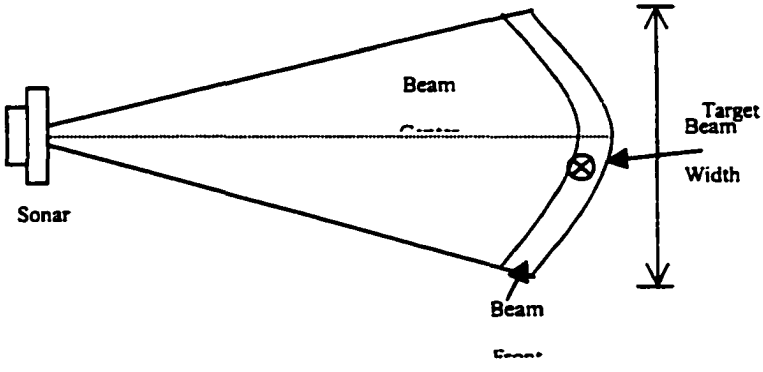

Figure 23: Ranging a Point Target

---

[36] The robotics community has two senses for the term 'specular.' The first refers to the theory that the range represents the shortest distance to a planar target. The second sense is to describe an environment in which the reflecting angles of multiple objects result in an indirect (multiply reflected) path for the sonar pulse. Rather than introduce redefinitions, we shall endeavor to make clear, by context, which sense is being employed.

[37] There is literature on (predictive) modeling the sonar response (e.g [93], [24]). Without very detailed information on the physical properties of the objects within the AOI, the sonar predictions are very inaccurate.

9.2 Hardware Implementation

The most common sonar sensor used in robotics, and the one we shall use in our research, is the Polaroid 6500 series Sonar Raging Module. [86] (See Figure 24.) This module (left side of Figure 24) consists of an analog chip (SN28784), a digital chip (TL851), and assorted discrete supporting components on a single printed circuit board. (The black disk on the right of Figure 24 is the 600 series transducer.)

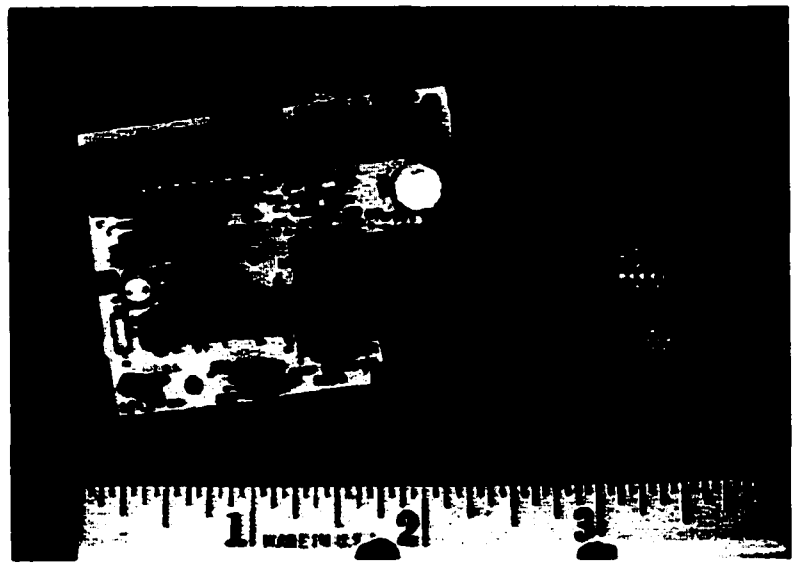

Figure 24: Polaroid OEM Kit (courtesy [8])

The interface to the module consists of a trigger line (INIT), a receive line (ECHO), a blanking line (BLNK), and a blanking inhibit line (BINH). When operated at 5 volts all signals are transistor-transistor logic (TTL). The cycle begins by bringing INIT high. This triggers the digital chip to send sixteen pulses through a transistor/transformer step-up circuit that raises the voltage of the pulses to the 400 volts required by the ultrasonic

transducer (see below). Once the signal is initiated, the receive circuitry is inhibited for 2.38 ms to eliminate false returns caused by transducer ringing. This blanking interval can be reduced by raising BINH high. The first returning echo is detected by the same ultrasonic transducer and passes through the same transformer, which now functions to reduce the voltage. The return echoes pass through the analog chip which performs, among other functions, step-wise amplification of the return signal. This receive gain is discretely incremented for 38 ms, corresponding to a detection range of 10 meters. The reason for this is to compensate for **spreading loss**:

$$\frac{I_{pulse}}{I_{echo}} = \frac{S_1}{S_2}$$

where I_{pulse} *and* I_{echo} are the acoustic intensities in watt/cm^2 of the pulse and echo, respectively, and S_1 and S_2 are the distances of the beginning of a specified ray path from the transducer and the distance of the target from the transducer, respectively.

and **attenuation loss**, which are the combined effects of scattering and absorption by the transmission medium (in this case, air). [50]

If an echo is detected during this interval, ECHO is raised high. It is the responsibility of the interfacing processor to time the interval between INIT and ECHO, and thus measure the target distance. If multiple target detection is desirable, BLNK can be raised to reset the ECHO line, allowing 0.44 ms for all sixteen pulses to return from the first target.

The 6500 module we use is connected to the Polaroid 600 series Instrument Grade Electrostatic Transducer.[85] When triggered by the 6500 module 400 volt pulse, the transducer produces a 50khz ultrasonic pulse. The same transceiver is used to received the echo returns. The nominal beam pattern has a primary lobe with approximately a 20° beam width, with primary side-lobes extending out another 15° -20 through –30 dB on either side of the primary beam. (See Figure 25.) In practice the 6500 module thresholds much of the side lobes to a detection/receive beam width of 22°-30°. Coupled with the 6500 module the detection range with default blanking is 2.4-1000cm. The ranging accuracy is stated to be ±1%.

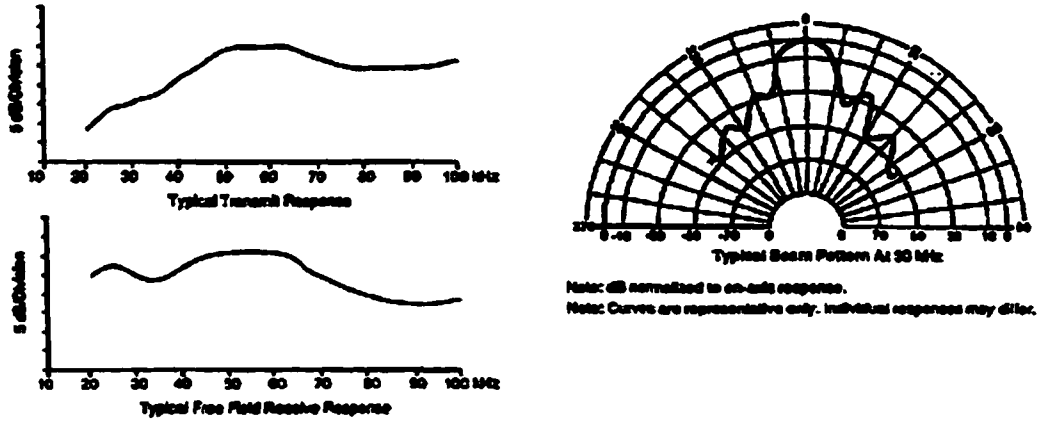

Figure 25: Polaroid 600 Series Transducer Beam Pattern [85]

The TTL level signals of the 6500 module are brought out to the general purpose I/O pins of the BX-24 micro-controller. This micro-controller consists of an 8-bit RISC

processor, 32 KB of EEPROM for program memory and byte code interpreter, 400 bytes of RAM, 8 channels of 10 bit analog-to-digital converter, plus 8 channels of general purpose I/O. The device is programmed via a standard RS232 interface, which can also be used to control the processor from a host computer while executing a program. The BX-24 is programmed in a dialect of BASIC that supports single precision floating point arithmetic, multi-tasking, semaphores, and queues.

The 600 series transducer is placed inside of an ABS plastic shroud, which is attached to a servo horn. The servo is mounted inside a plastic project box, which allows the transducer to be rotated approximately 180°.

9.3 Software Implementation

The controlling software, residing completely within the BX-24 controller, is designed to support both data gathering and sensor fusion. Thus the interface layer must both support the virtual sensor abstraction described in Chapter 4, and also a human readable interface to a modem program (PCPLUS) executing on the host computer. The commands described in Table 3 both support hand-shaking and query-response interfaces to support a fusion algorithm executing on the host computer (**rotate/ping/data/calibrate**) and a command for collecting large amounts of raw sensor data on the host computer (**repetitive ping with immediate display**).

Command	Syntax	Return Parameters
rotate	r,<degrees>,	command acknowledged = CR command rejected = e
get data vector	d,	<data vector> command rejected = e
calibrate	c,	command acknowledged = CR command rejected = e
ping	p,	command acknowledged = CR command rejected = e
repetitive ping with immediate display	P,<number of pings>,	<data vector> command rejected = e

Table 3: Sonar Command Syntax

The **rotate** command rotates the servo between 0° (far right) and 180° (far left). The **get data vector** command returns data collected during the most recent **calibrate** or **ping** where range is in millimeters and angle is in degrees. **Calibrate** takes a series of measurements (currently 25) and stores the zero mean sample variance calculated by equation (8.3). **Ping** takes a single measurement, but does not return the data vector. **Repetitive ping** takes a series of measurements, returning the data vector following each measurement. This last command is used to process sensor data off-line, while **ping** and **get data vector** are used for sensor fusion.

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \qquad (9.3)$$

where

s^2 is the sample variance

n is the sample size

$\bar{x}$ is the sample mean

9.4 Test Data

In any laboratory experiment, it is important to know the characteristics of one's test equipment. For the sonar range finders, the first characteristic to be determined is the ideal resolution. In this case the resolution is limited by the micro-controller real-time clock, which is 1.0μs. Using equation (9.2), where γ = 1.4, R = 286.9 N/Kg, and T = 295.16, the speed of sound is 344.0m/s. Under these conditions, in 1.0μs sound can travel 344.0 * 1.0E-06 ≅ 3.44E-04 mm. Using equation (9.1), the resolution of the sonar is no better than approximately 0.0688 mm.[38]

With these unreasonably unrealistic expectations, the next step is to determine the ranging accuracy empirically. In this experiment each sonar is placed a set distance from the target. The target is 14cm×14cm square board mounted to an adjustable tripod. The

---

[38] This does not take into account the time lag between the micro-controller issuing the ping command to the 6500 module, and the micro-controller detecting the return echo. This lag is assumed to be a constant.

board is perpendicular to the floor. The bottom edge of the target is 100cm from the floor. The tripod is wrapped with acoustic damping foam in order to minimize stray echoes. Range readings are made at the distances: 100cm, 150cm, and 200cm. Assuming a beam width (conical) of 22° with the transducer aimed at and parallel to the center of the target, neither the ceiling nor the floor will produce an echo before the target.

The first set of tests were designed to calibrate the sonars. Using measured ranges to the target, the ranging accuracy was 2%-3% short for all sonars at all three ranges. This is within the expected environmental variance (due to humidity and other factors) of 2% [24] plus 1% ranging inaccuracy of the 6500 module [85].

The next set of tests was to determine the measurement variance. Five hundred samples were made of the target at each one of the distances. The results appear Table 4.

Sonar	Nominal Range	Mean Measured Range	Variance
1	1000mm	1000.0	0.022
	1500mm	1499.0	0.048
	2000mm	2001.0	0.076
2	1000mm	1001.0	0.02
	1500mm	1499.0	0.043
	2000mm	2001.0	0.029

Table 4: Range Measurements

From these measurements it appears that measurement variance does not necessarily correlate with range. This simplifies calculations: it appears that a pessimistic variance of 0.076 will apply across sensors of this type with the ranges (100cm-200cm) measured.

The last question to be answered concerns the distribution of the measurements about the mean. Much of the literature and many of the simple models assume this distribution to be Gaussian. Is this assumption warranted? Figure 26 shows histograms (buckets = 10) for sensor 1 (top row) and sensor 2 (bottom row) at all three distances ($f(x)$), plotted against a normally distributed random vector ($g(x)$) with the same mean and variance.

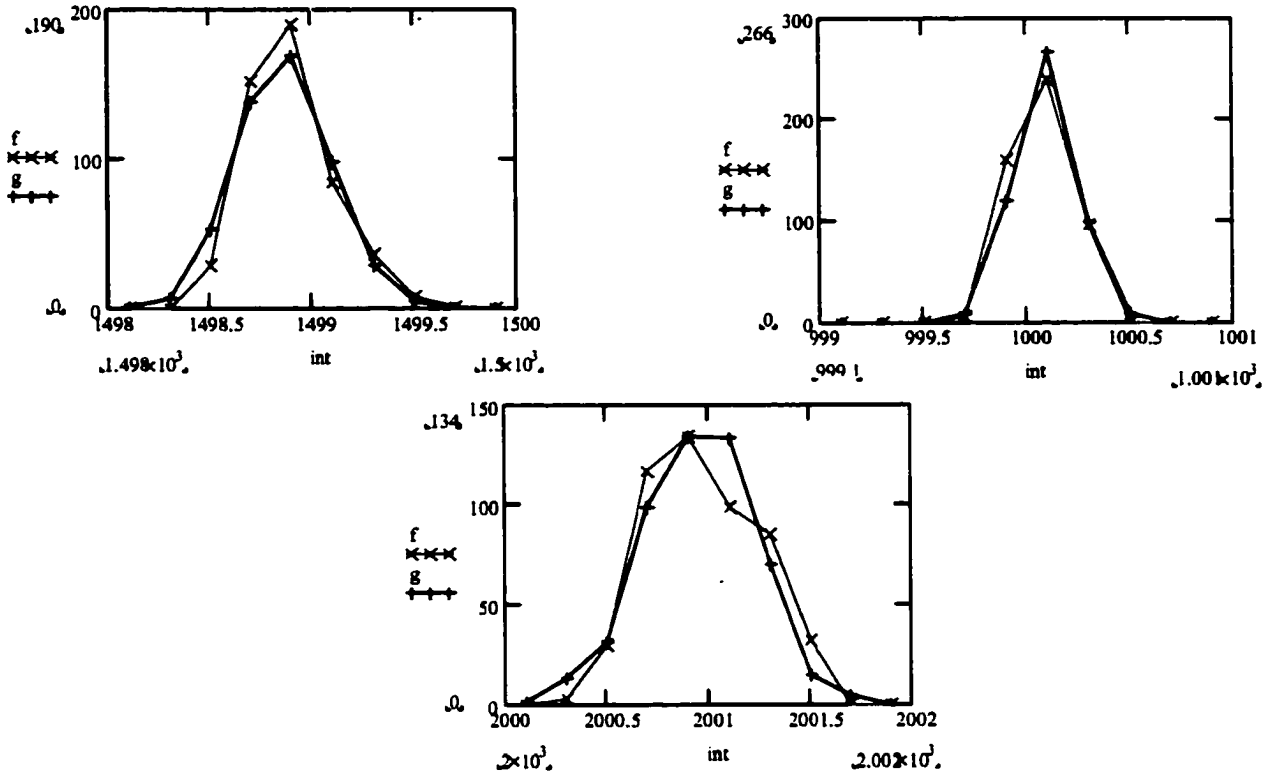

Figure 26: Measured Ranges vs. Normal Distribution

These results correspond with those in [24], where it is asserted that the single target/single sensor case can be modeled by a normal distribution.

A final, and rather parenthetical note to the results described above: As pointed out in [24], the normal distribution is only a reasonable approximation in a non-specular environment. The sensor must have a clear shot at the target, and the target itself must reflect a fairly strong and consistent waveform back to the transducer. In a number of trials the sensor was located too close to the floor, or a wall, such that the transducer

picked up echoes other than those reflected immediately and directly back from the target. This was immediately apparent when analyzing the data; the data histogram deviated significantly from a normal distribution. One example from 150cm is shown in Figure 27 below.

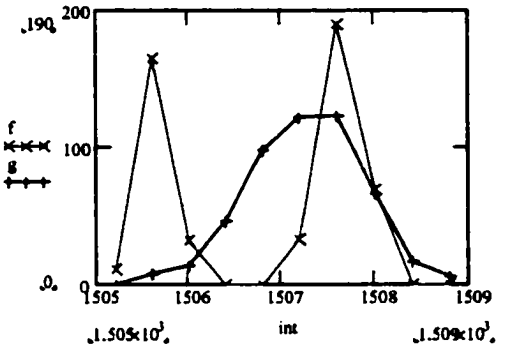

Figure 27: S1 Specular Histogram

The histogram shows two distinct peaks: one at approximately 1505.5mm, and the other at 1507.8mm. An interesting research project would be to determine if the sort of distribution depicted in Figure 27 is a reliable marker for specular environments and, if so, whether it can be used as a parameter in computing a reliability metric for the fused measurement.

Appendix B: A Virtual Sensor

The crucial experiment designed to test the hypothesis proposed by this thesis (as described in Chapter 6) dictated the design of the virtual sensor. First, the data vector returned by the sensor must provide target location in two-space relative to the sensor. A second consideration was that, in order to fuse this data vector with others, the data must be commensurate. Thirdly, though the sensor would be fixed, and not located on a mobile platform, it should adaptable to an autonomous mobile robot. These considerations pointed to a **vector sonar**: one which, unlike the standard Polaroid 6500 module in common use in robotics, returns both the target azimuth and the target range

This appendix consists of three sections. First, the geometry of the virtual sensor is described. Then we describe the hardware and software implementations of the sensor. Finally we examine some of the properties of this sensor.

10.1 Sensor Geometry

The operation of the Polaroid 6500 module is described in more detail in Appendix A. In short, the transducer emits a conical beam of sound at 50Khz. This beam reflects off targets within the beam width, which returns to the transducer to be amplified

and recorded. The response pattern of the transducer depicted in Figure 28 extends approximately 11° to either side of the beam center.

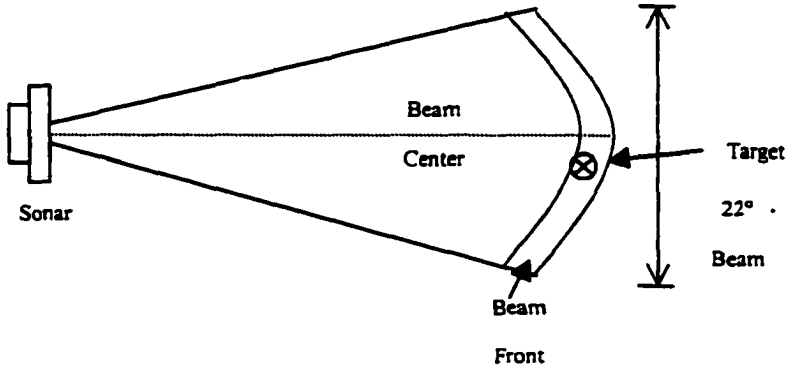

Figure 28: Ranging a Point Target

As the beam front moves from the transducer it is partly absorbed, partly reflected by any object in its path. Generally we record only the first such reflection since subsequent reflections tend to be specular. Thus the range returned is from the sonar to the nearest **point of an object** that lies within the 22° beam width. The bolded phrase in the previous sentence is critical: if the sonar is used to illuminate a wall, for example, the range returned will generally be the closest point of the wall within the beam width. Thus if the wall is at an angle, as shown in Figure 29, the closest point (and thus the range returned by the sonar) will actually be at the edge, and not the center of the beam.

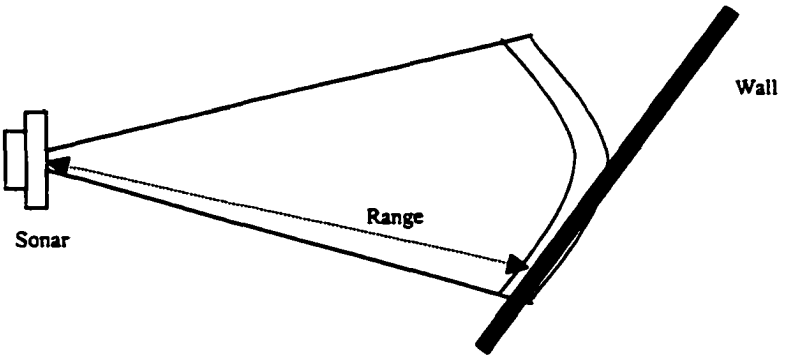

Figure 29: Ranging an Oblique, Planar Object

As noted in [10] and subsequently [59] and [12], a pair of sonar transducers mounted in a parallel array can be used to extract more information than the single sonar in either the point target (Figure 28) or planar target (Figure 29) example. The geometry of the point target/sonar array is depicted in Figure 30.

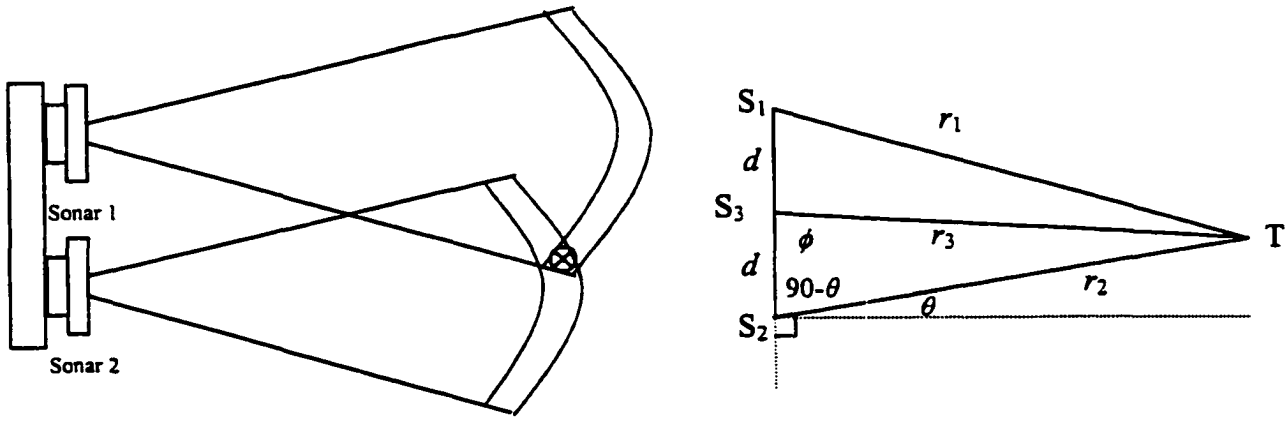

Figure 30: Virtual Vector Sensor

The target in the left diagram of Figure 30 lies within the frontier of Sonar 1 (S_1) and Sonar 2 (S_2), separated by a distance of $2d$. The virtual sensor S_3 lies midway between S_1 and S_2. Within a field of width equal to the beam width of either sonar (i.e., 22°), through the use of some simple trigonometry, the target azimuth (ϕ) and range (r_3) relative to S_3 can be calculated. The known values are d, r_1, and r_2. The azimuth of T relative to S_2 (θ) is easily calculated from the Law of Cosines:

$$\theta = \sin^{-1}\left(\frac{(2d)^2 + r_2^2 - r_1^2}{4dr_2}\right) \tag{10.1}$$

r_3 can also be calculated from 90-θ by Law of Cosines:

$$r_3 = \sqrt{d^2 + r_2^2 - 2dr_2\cos(90 - \theta)} \tag{10.2}$$

Finally, azimuth (ϕ) can be derived from r_3 by substitution in equation (10.1):

$$\phi = \sin^{-1}\left(\frac{(d)^2 + r_3^2 - r_2^2}{2dr_3}\right) \tag{10.3}$$

where

$$-11^\circ \leq \phi \leq 11^\circ$$

relative to virtual sensor S_3.

The virtual vector sensor is useful in certain limited circumstances. The azimuth accuracy for a single point target, proportionate to d, is far greater than that provided by the 22° cone of the single sensor. There are a number of members of this equivalence class, however, that provide false readings.[39]

[39] See Section 4.1 for an explanation of equivalence classes.

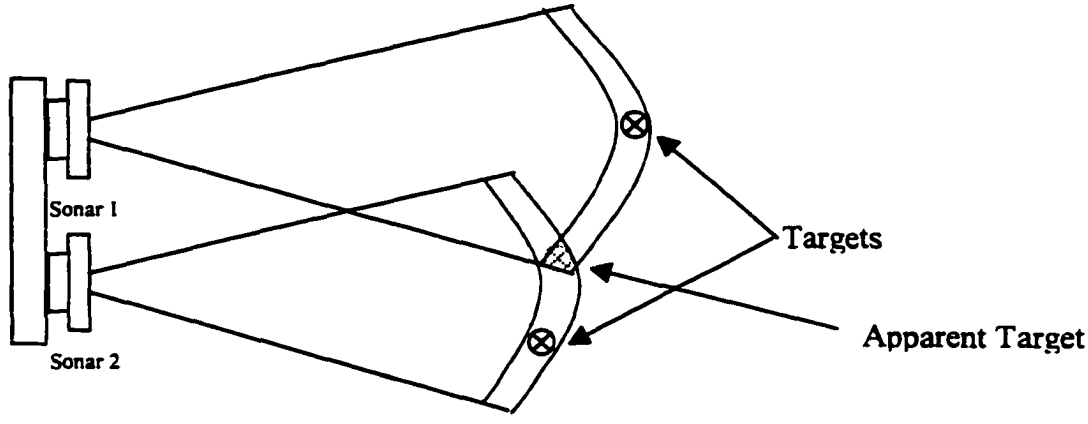

Figure 31: Equivalence Class

Figure 31 depicts a case where two targets give the illusion of a single target in the overlapping frontiers of the two sensors. The oblique planar target in Figure 29, of course, would also be a member of this equivalence class.

If there is reason to believe that the target is a planar object (such as a wall), the sensor array can easily be configured as a virtual feature sensor. In this case, instead of providing a target azimuth, the virtual sensor returns the orientation of the feature relative to the array. The left diagram in Figure 32 depicts the array configured as a virtual feature sensor, while the right diagram depicts the trigonometric relationships.

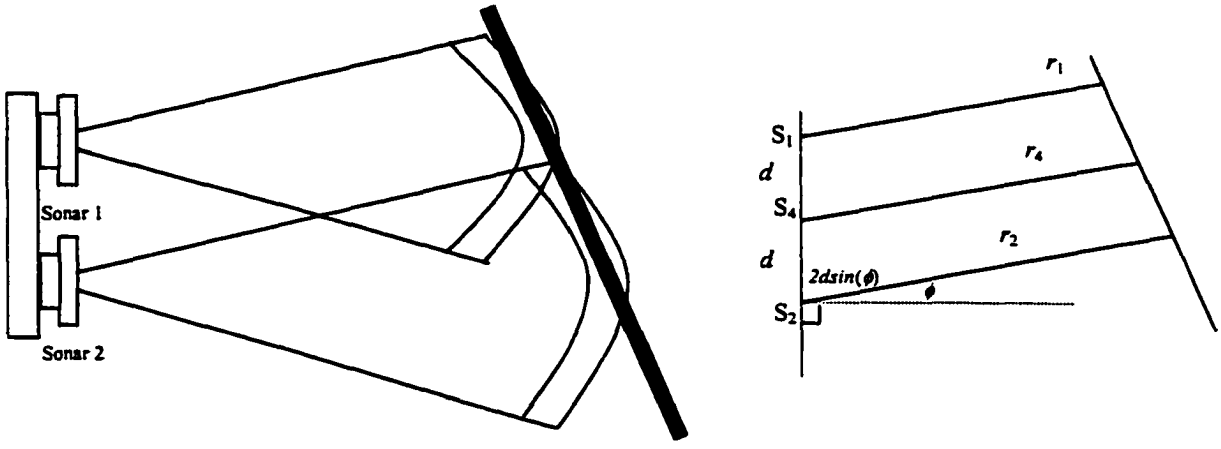

Figure 32: Virtual Feature Sensor

$$\phi = \sin^{-1}\left(\frac{r_1 - r_2}{2d}\right) \text{[59]}$$

where S_4 is the location of the virtual sensor as before, and r_4 is simply

$$r_4 = \left(\frac{r_1 + r_2}{2}\right).$$

Just as was the case with the vector sensor, the feature sensor admits of a significant equivalence class. The examples depicted in Figure 30 and Figure 31 would appear, if interpreted by the feature sensor, as oblique walls. Given the shortcomings of the primary sensor (the sonar), there must be sufficient prior evidence that the target is a single point target, or a single continuous wall, before the data from either virtual sensor can be used with any form of confidence.

10.2 Hardware Implementation

The hardware implementation is similar to that of the single sonar, with addition of a second transducer (600 series) and ranging module (6500 series).[40] The transducer shrouds are mounted on either ends of an aluminum bar stock arm. The middle of the arm is attached to a servo horn. The servo itself is mounted inside a small project box, which is attached to an adjustable tripod. (See Figure 33.) The control lines for the transducers run to a breadboard on which are mounted the two 6500 modules and the sensor micro-controller. The sensor micro-controller also provides the pulse-width modulation (PWM) signals to drive the servo motor. The servomotor allows the transducer arm to be rotated approximately 180°. The micro-controller accepts commands and returns data via a single serial port.

In operation, each sonar is triggered independently, after a sufficient delay to ensure neither picks up the other's returning echo. We experimented briefly with Whiteside's design for a passive sonar. This design uses a NAND gate and a pair of output pins from the micro-controller to disable the ping from one transducer (passive) while allowing it to receive the echo from the second transducer (active). We dispensed with this design for several reasons. First, if the interval between either active sonar's ping is

---

[40] For details see Appendix A.

sufficiently short relative to the target velocity, the passive/active sonar design is indistinguishable from using two active sonars. Secondly, the sensor micro-controller used has only a single hardware timer, and can only provide high resolution input timing from one pin at a time. Thus the passive/active sonar design was required to alternate sonars anyway.

Figure 33: Sonar Array

10.3 Software Implementation

The software implementation is similar to that for the single sonar described in Appendix A. Four sonars are defined: left (S1), right (S2), vector (S3), and feature (S4).

Two additional elements are defined for the S3 and S4 data vectors: target azimuth and azimuth variance, and target orientation and orientation azimuth respectively. Table 5 depicts the command syntax.

Command	Syntax	Return Parameters
rotate	r,<degrees>,	command acknowledged = CR command rejected = e
get data vector	d,<sonar number>,	S1 <data vector> S2 <data vector> S3 <data vector> S4 <data vector> command rejected = e
calibrate	c,<sonar number>,	command acknowledged = CR command rejected = e
ping	p,<sonar number>,	command acknowledged = CR command rejected = e
repetitive ping with immediate display	P,<sonar number>,<number of pings>,	S1 <data vector> S2 <data vector> S3 <data vector> S4 <data vector> command rejected = e

Table 5: Sonar Array Command Syntax

The **rotate** command rotates the servo between 0° (far right) and 180° (far left). The **get data vector** command returns data collected during the most recent **calibrate** or **ping**. **Calibrate** takes a series of measurements (currently 25) and stores the zero mean sample

variance calculated by equation (10.4). **Ping** takes a single measurement, but does not return the data vector. **Repetitive ping** takes a series of measurements, returning the data vector following each measurement. This last command is used to process sensor data off-line, while **ping** and **get data vector** are used for sensor fusion.

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \qquad (10.4)$$

where

s^2 is the sample variance

n is the sample size

$\bar{x}$ is the sample mean

10.4 Test Data

The individual components of the vector sensor (S3) and feature sensor (S4) are the two sonars (S1 and S2) described in Appendix A. The test data are different, however, since now we alternate sonars with approximately a 0.3 second pause between each ping. The readings are made at the three distances: 100cm, 150cm, and 200cm, at the same target as in Appendix A. Table 6 shows the range measurements for S3, with the target being located approximately at 0° (straight ahead).

Sonar	Nominal Range	Mean Measured Range	Variance
1	1000mm	1002.0	0.038
	1500mm	1501.0	0.065
	2000mm	2001.0	0.088
2	1000mm	1003.0	0.044
	1500mm	1503.0	0.06
	2000mm	2003.0	0.106
3	1000mm	999.547	0.031
	1500mm	1500.0	0.053
	2000mm	2000.0	0.086

Table 6: S3 Range Measurements

Some observations on Table 6: First, the measurement variances once again do not correlate to distance. Second, the mean measured range for S3 is less than either S1 or S2 because the feature sensor lies at the midpoint between S1 and S2.

Next, Figure 34 contains three histograms corresponding to each of the three ranges (S3 only), plotted against a normal distribution.

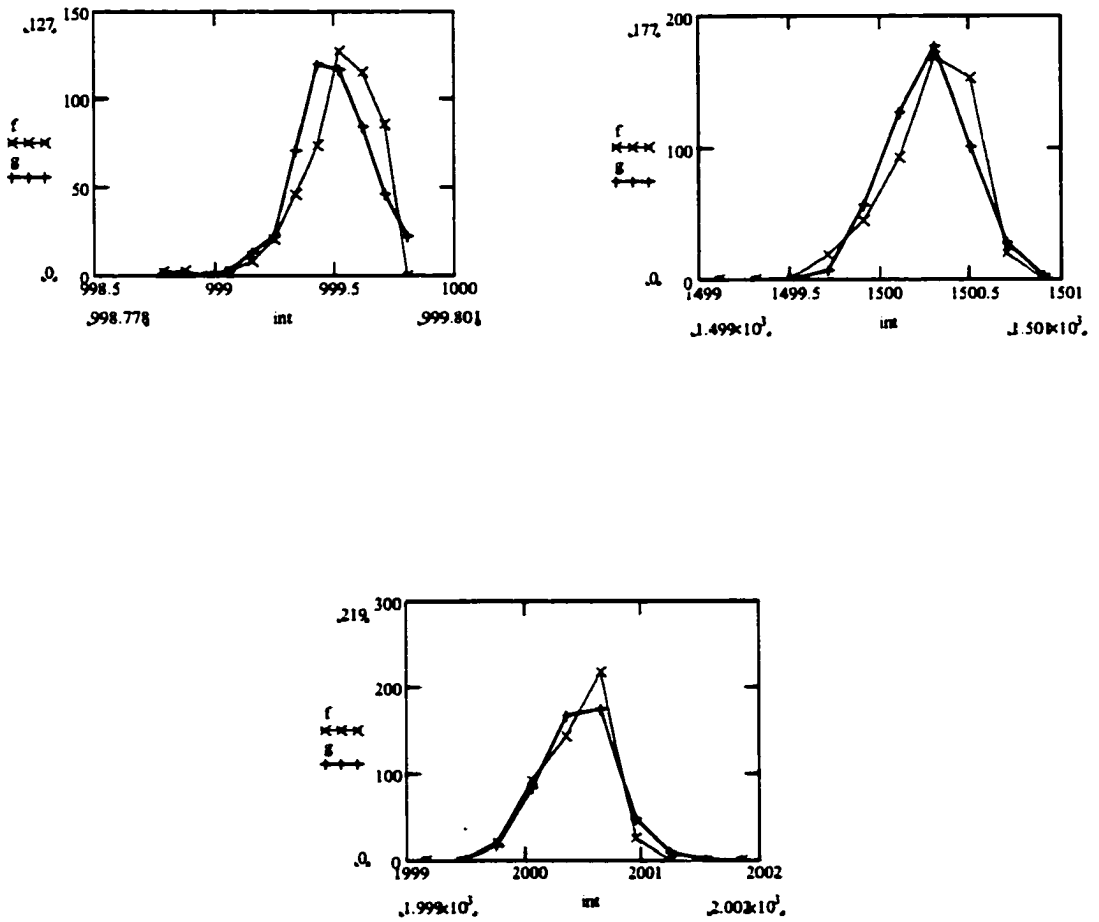

Figure 34: S3 Measured Ranges vs. Normal Distribution

The vector sensor data vector includes both a range element and an azimuth element. Since the accuracy of the azimuth is purely a function of the component ranges (equations 11.1 – 11.3), it makes little sense to calibrate this parameter independently. The measured variance in the azimuth for all three ranges (100cm, 150cm, and 200cm) appears in Figure 35. The x axis on these graphs represents degrees, where 0 indicates that the target is directed ahead of the sensor (i.e., perpendicular to the sensor arm). Once again, $f(x)$ is the measured azimuth, while $g(x)$ is a normal distribution.

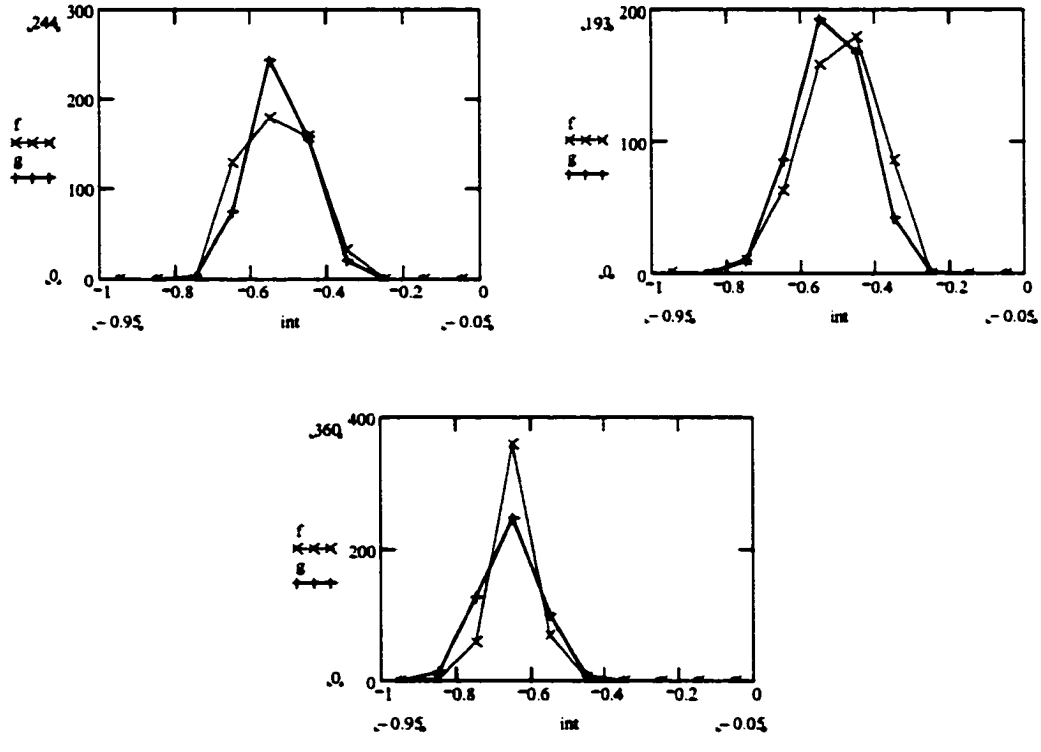

Figure 35: S3 Computed Azimuth

Table 7 depicts the variances in the computed azimuth for each of the measurement ranges: 100cm, 150cm, and 200cm.

Nominal Range	Azimuth Variance
1000.0mm	0.031
1500.0mm	0.053
2000.0mm	0.086

Table 7: S3 Azimuth Variances

Finally the same measurements were made with the feature sensor (S4). This sensor, as described above, computes the angular relation of a planar object in respect to the sensor arm. Each set of readings was made with a computed angle of approximately 3°, where 0° represents a wall parallel to the sensor arm.

Nominal Range	Angular Variance
1000.0mm	0.04764
1500.0mm	0.077
2000.0mm	0.05674

Table 8: S4 Angular Variance

Figure 36 presents the histogram of the computed angles for each of the three distances. These data indicate that, just as is the case with a single sonar (S1 or S2), and assuming a non-specular environment, it is reasonable to model the vector sonar (S3) or the feature sensor (S4) as a Gaussian random variable.

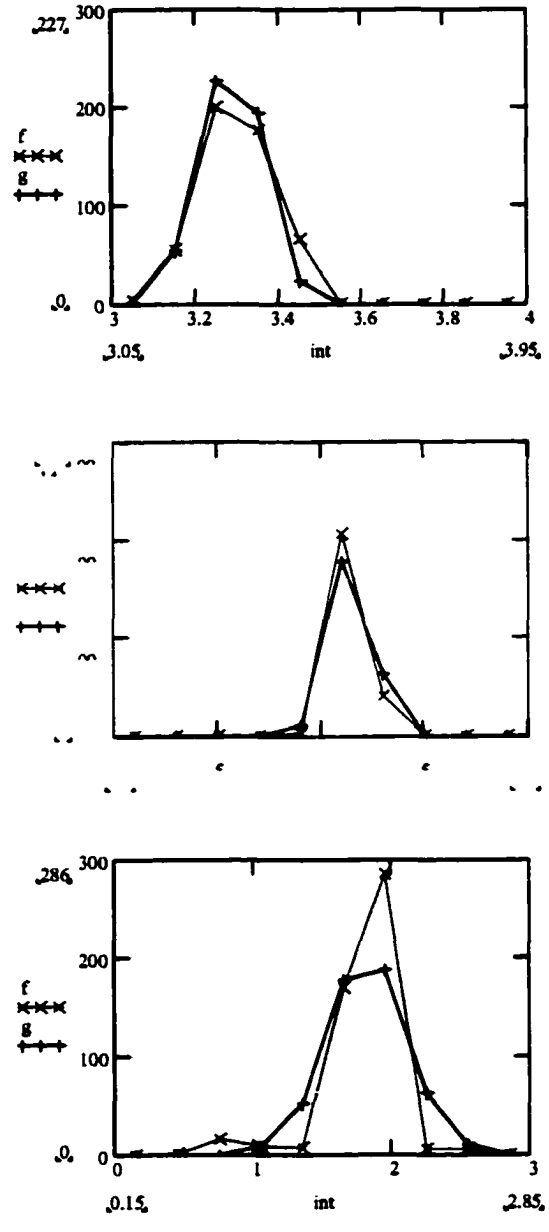

Figure 36: S4 Computed Relative Angle

Appendix C: Integrated Laser/Sonar Range Finder

The third sensor is a combination of a laser line scanner and a sonar range finder. The first section provides background and theory behind this integrated sensor. The next two sections describe implementation and testing/calibration, respectively.

11.1 Theory

Appendix A describes the positive and negative features of the sonar – on the plus side, under ideal circumstances, the range measurement is extremely accurate. On the negative side specular responses can produce wildly inaccurate readings (e.g. [51] and [65]). Even in fairly sparse environments, the sonar beam width provides only a very coarse estimate of target azimuth. Appendix B describes one method for dealing with this second problem: the vector sonar array. Like the single sonar sensor the array provides very precise ranging estimates in non-specular environments. Unlike the single sonar it can, given a target that lies within the beam widths of both transducers, provide a very accurate target azimuth. The major weakness of this approach is the inability to distinguish between a single target lying within both transducers' cones, and two targets in each cone.

The solution to the sonar problem would seem to be a ranging sensor with (1) a narrow beam width and (2) minimal specular properties. Enlarging the transducer would result in a narrower beam width, since the beam width is inversely proportional to the transducer diameter. Some work has been performed on detecting specular responses by analyzing the echo waveform [65]. This might allow filtering of specular responses, but is probably of little use in a highly specular environment.

Laser range finders, or LIDARs, have become very popular in robotics due to their precise ranging accuracy, narrow beam width, and relative immunity from specularity. An excellent example can be seen in [8][41], which is particularly telling when compared with a comparable sonar scan [65].[42] LIDARs, unfortunately, are still very expensive. In addition to this, as pointed out in [97], the virtues of a LIDAR are also its chiefest vices: with such a narrow, collimated beam, scanning must now be performed in two dimensions rather than one. In a single pulse the sonar returns data from a 20-30° cone; a LIDAR must line scan the same region. Since this is currently performed mechanically[43], LIDARS are far slower to detect the nearest object than sonars are. Thus [109] integrates both sensors, using the LIDAR to disambiguate the sonar return.

---

[41] Figure 4.33.

[42] Figure 2.

[43] Millimeter wave imagers are able to scan electronically.

Moving away from time of flight range finders, there is a large class of ranging sensors that depend on stadiametric measurement to determine range. One method is to use a pair of cameras, and to calculate depth based on image disparity. [18] A much simpler (both in terms of hardware and computation) is to project a light onto the target a fixed distance from the camera. [107] [72] Typically a laser line-striper is used, due to the minimal beam divergence. Figure 37 depicts the geometry of the line-stripe range finder.

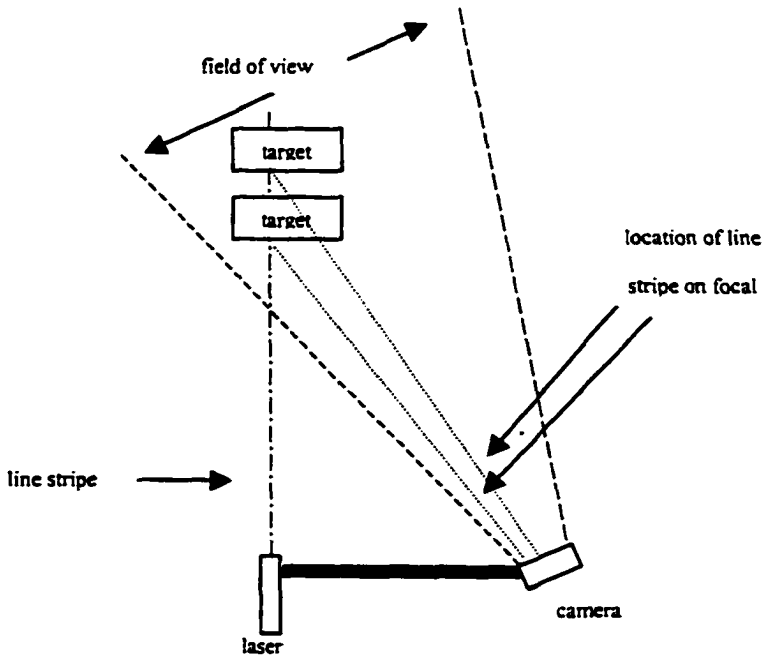

Figure 37: Line Stripe Geometry

The laser line is registered to the center of the camera field of view at some known distance. As the target moves directly toward the laser, the line moves across the focal plane from the center to the left edge. As the target moves away from the laser the line moves toward the right edge. Since the laser is perpendicular to the sensor arm the target azimuth is simply the sensor pose.

The angular resolution of this sort of sensor is dependant upon the length of the arm supporting camera and laser, and the resolution of the camera. The Sharp GP2D02, for example, is a very inexpensive (<$30) implementation of this principle, using a narrow beam diode in place of the laser, and a photo-optical grid instead of a camera. The performance is correspondingly limited: 80cm range and 100mm beam width. [100]

Calculating range based on the movement of the line stripe across the camera focal plane involves two considerations. First, as an object moves away from the camera, its image occupies less and less of the focal plane. Thus as the target moves away from the camera at some constant rate, the movement of the line stripe across the focal plane slows proportionately. So in order to calculate the relationship between pixel column and line stripe distance, one first has to calculate the camera field of view (FOV) as a function of distance. This equation is given in equation (11.1), where A is the camera CCD array size, D is the distance from the lens to the target, and f is the focal length of the lens.

$$D = f\left(1 + \frac{FOV}{A}\right) \qquad (11.1)^{44}$$

As stated below, the focal length of the camera used here is 38mm. As the documentation does not provide the array size, we have used (11.1) to determine A to be approximately 39.912mm. This is fortuitous. By plugging this into (11.1) it can easily be seen that the FOV is roughly equal to the distance from lens to target.

The second consideration is the apparent lateral movement caused by the angle between the camera and the laser.

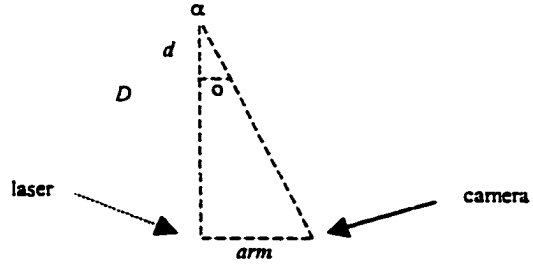

Figure 38: Lateral Displacement

In Figure 38 α represents the angle between the camera center of field and the laser line when registered on a target 1500mm from the laser; *arm* is the distance between the laser and the camera center of field; D is the distance to the registration point (1500mm), and d is the distance from the registration point to the target; and finally o is

[44] http://www.vision1.com/systems/fovmath.html

the apparent displacement of the laser stripe to the left of the center of the camera field of view. The equation for calculating o appears in equation (11.2).

$$o = d\tan(\alpha) \tag{11.2}$$

Once o has been calculated, the next step is to determine the percent of the FOV taken up by o. This is calculated in (11.3).

$$percent = \frac{d\tan(\alpha)}{FOV} \tag{11.3}$$

Finally, multiplying *percent* times the number of columns in the CCD matrix, and then adding this to the CCD midpoint, gives the column number as a function of range. We make the substitution $d = D - r$, where r is the range to the target.

$$column\# = \frac{\#\_of\_columns}{2} + \left\{\frac{(D-r)\tan(\alpha)}{FOV}\cdot \#\_of\_columns\right\} \tag{11.4}$$

Note that where $D = r$ (the target range equal to the registration point), then the column number is the mid-point in the CCD array. Because of (11.1) *FOV* for the QuickCam can be simplified to r.

Equation (11.4) is not very useful in and of itself; we need the range as a function of column number, not the inverse. The inverse of (11.4) is fairly straight-forward, and appears in equation (11.5).

$$r = \frac{D\tan(\alpha)}{\tan(\alpha) + \frac{column\_number}{\#\_of\_columns} - \frac{1}{2}} \tag{11.5}$$

A plot of (11.5) given D = 1500mm appears in Figure 39. The y-axis is the range in millimeters, while the x-axis is the column number. As a rule of thumb, the greater the slope, the greater the imprecision. For example, column 130 to column 131 represents a gap of 140.9mm, while column 180 to column 181 represents a gap of only 14.676mm. Also note the function of D and α: since α is a function of D and the distance between the camera and the laser, precision would be greater if the camera were positioned a greater distance from the laser.

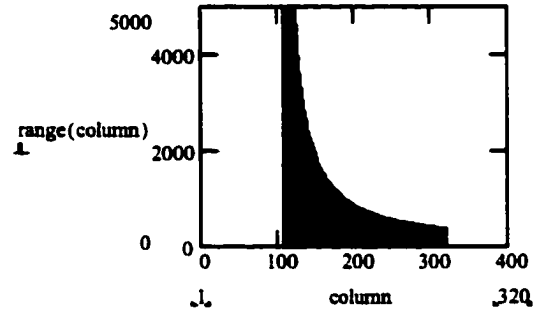

Figure 39: Laser Range Finder (Column Number vs. Range)

The above paragraphs describe how to compute two of the parameters for the measurement vector: target range and target azimuth. For each of these parameters the measurement precision also has to be calculated.

For the purposes of simplification, represent equation (12.5) as $r = f(c)$, where c = column number and r = the target range. Precision can be represented by the scalar $r_{precision}(c) = |f(c+1) - f(c-1)|$. Note that, unlike the sonar described in Appendix A and 2, this is a variable parameter dependent on range to target. This is illustrated Figure 39, where the slope of the curve increases as the target range increases.

Assuming very accurate segmentation, azimuth precision is also computed by $\phi_{precision}(c) = |g(c+1) - g(c-1)|$, for some function $g(\cdot)$. To compute this function, first note that the interval $(c-1, c+1)$ represents $\left(\frac{3}{320}\right)FOV \approx 0.009375r$. The angle of the right triangle formed by the sensor, the target, and a point $0.009375r$ to one side of the target is:

$$\tan^{-1}\left(\frac{\left(\frac{3}{320}\right)\left(\frac{1}{2}\right)r}{r}\right) = \tan^{-1}\left(\frac{3}{640}\right)$$

The complete interval is twice that, so the precision is:

$$\phi_{precision} = 2\left(\tan^{-1}\left(\frac{3}{640}\right)\right) = 0.269^{\circ} \quad (11.6)$$

Note that, unlike range precision, azimuth precision is a constant.

11.2 Implementation

The line-striping range finder consists of two major components: a laser line striper and a digital camera.

11.2.1 Hardware

The camera used for the line striper is the Connective QuickCam monochrome camera. The camera itself is a charge coupled device (CCD) array, consisting of 336 by 243 pixels. The first twelve columns are used for calibration, so the image area is 324 by 243. The CCD array is capable of four or six bit grayscale modes. The analog signal from the CCD is sent to an analogue to digital converter (ADC), which is then sent to the host computer parallel port. Mounted in front of the CCD array is an infrared filter, which is required due to the sensitivity of the CCD array to near infrared. The camera lens is equivalent to a 38mm lens, with an f-stop of f1.9. The device is attached via a parallel port plug (which contains much of the signal processing circuitry) to the computer parallel port. A second connector is attached in-line with the host computer key-

board in order to tap into the 5-volt power, with a peak power requirement of 120mA. [25]

The QuickCam interface consists of commands sent from the host computer to the QuickCam, and image data returned from the camera to the host computer. To transmit commands to the camera the interface software writes to PORT+OFFSET, where PORT = {0x3bc, 0x378, 0x278} for ports lpt0, lpt1, and lpt2 respectively, and OFFSET = 0 for the device data port. Since this port is eight bits wide, all commands are **unsigned char**. There are four more parallel port output lines defined by IEEE 1284-A [91], which correspond to pins 1, 14, 16, and 17 (see Table 9), which are also used to control the camera. These pins map onto the eight bit parallel port control register located at OFFSET = 2.

Name	Pin Number
Strobe	1-
DB0	2
DB1	3
DB2	4
DB3	5
DB4	6
DB5	7
DB6	8
DB7	9
ACKIN	10
BUSY	11-
PARAMEND	12
SELECT	13
Autofeed	14-
Error	15
Initialize	16
SELECTIN	17-
Ground	18-25

note: (-) indicates an inverted signal

Table 9: Parallel Port Pin Assignments

Retrieving data from the QuickCam is more complex. The standard parallel port (SPP) was not designed for bi-directional data transfers. When attached to an SPP, the QuickCam communicates in nybble mode, which means that it transfers one half of each pixel via the parallel port input status lines 10, 11, 12, and 13, which map onto the parallel port status register at OFFSET = 1. For an enhanced parallel port (EPP) or an extended capabilities port (ECP), full eight bit bi-directional transfers are supported.[45]

The line-striping laser is a 5VDC, 4mW, 660 nM laser module with a 4° line generating lens. The laser is attached to one end of a 30 cm arm; the camera is attached to the other. The distance between the camera lens and the laser lens is 25cm. The arm itself is mounted to a servomotor, which allows precise azimuth positioning.

11.2.2 Software

The software used to capture data from the QuickCam and render it as an $n \times m$ bitmap, is provided by the public domain **qcam** software. The software allows selection of 4 bits-per-pixel (bpp) and 6 bpp grayscale, at varying resolutions, with correspondingly diminishing performance in terms of frames per second (fps). All experiments were conducted at 320 × 240 resolution with 6bpp grayscale. The **qcam** software also provides controls for setting the camera contrast, which set the analog amplitude of the CCD pixel before it reaches the analogue to digital converter.

---

[45] Additional information on the QuickCam is available on a non-disclosure basis.

For software development, calibration, and initial testing, we use a software suite developed from the public domain **xfqcam** package. The extensive modifications made to **xfqcam** are described in the dissertation proposal; suffice it to say here that these modifications simplify the task of scripting and testing image-processing algorithms in a highly modular fashion. In operation, however, all display code has been removed, and the image bitmap is directly processed by the segmentation and range finding algorithms.

The input to the laser range-finding process is an image consisting of the target (or portion thereof), numerous other objects within the camera field-of-view, and the vertical line stripe. The output of the process is the pixel column number in which the stripe appears. As described above, it is then possible to calculate the range to the object and the target azimuth.

The problem remaining is how to determine the pixel column of the laser stripe. The first processing step is to remove everything from the image but the laser stripe. A computationally cheap method would be to take two images, in rapid succession, one with the laser on and one with the laser off, and then to XOR the images. When this is done with the QuickCam, it is obvious that there is substantial noise from frame to frame. Subsequent image processing performs some simple segmentation by first using a partial difference operator [23], and then taking a contrast histogram to determine which column has the predominance of the bright pixels. Figure 40 depicts the complete image with the

laser stripe in the center; Figure 41 shows the result of this image filtering/line segmentation.[46]

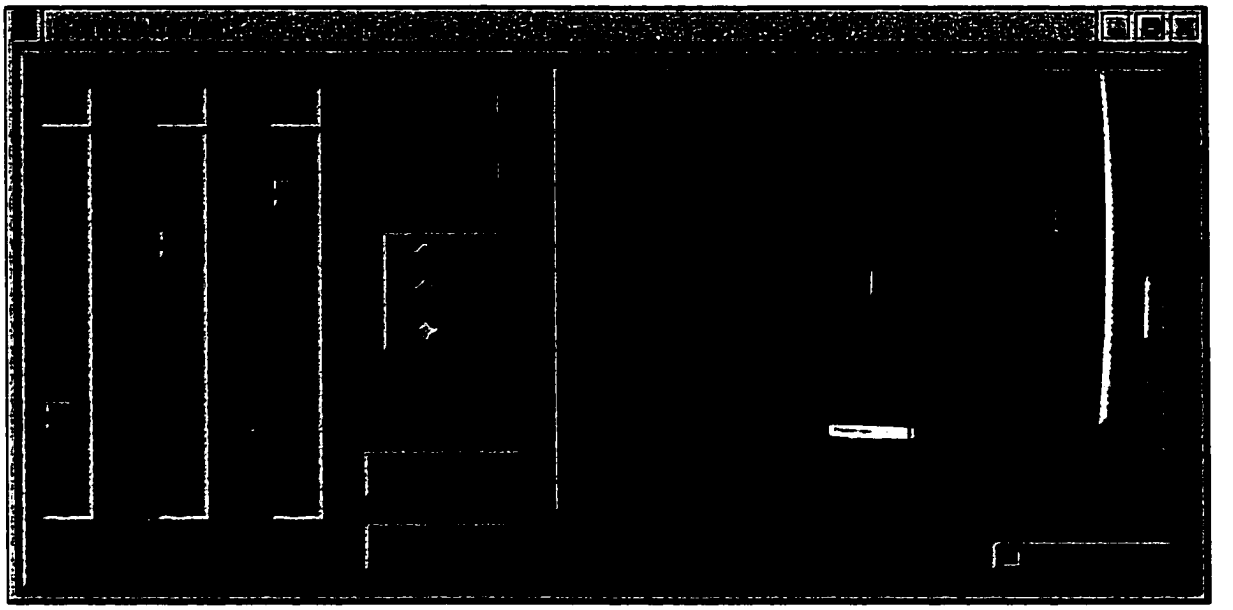

Figure 40: Image with Laser Stripe

[46] Segmentation is not the ideal approach for line extraction. Many infrared ranging sensors, such as the Sharp GP2D02 described above, pulse the source illumination at some fixed frequency (such as 40khz), and then pass the returned signal through a narrow band-pass filter.

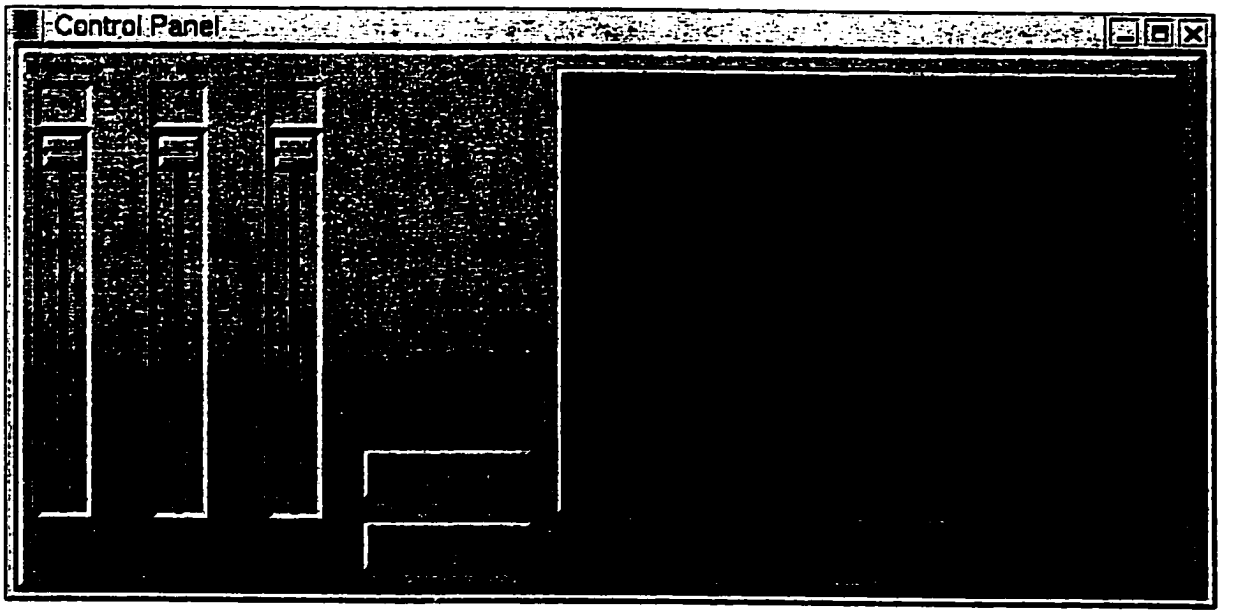

Figure 41: XORed Image

Once the column number is extracted (11.5) is used to calculate the target range.

Appendix D: Virtual Sensor – Implementation

As described in Chapter 1, there are two parts to the virtualization/modularization process: sensor virtualization and algorithm wrapping. This appendix represents initial experiments conducted on the former.

12.1 Control Vector, Data Vector

The virtual sensor module specification consists of four elements. Since the architecture is "data pull" rather than "data push," the inputs to the module are either control inputs or requests for information.[47] The outputs from the module are either acknowledgements in the case of a control input, or the data vector in response to a request for information. Figure 42 is an illustration of this design.

---

[47] A "data push" architecture is where the information source reports, unsolicited, to the information subscriber. As described in Appendix A, the **P** command (used for off-line analysis), is actually a "data push" design.

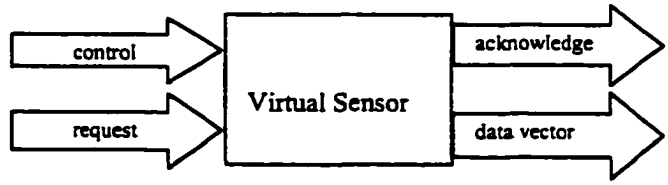

Figure 42: Virtual Sensor Specification

The data vector consists of two sub-vectors: the first is the sensor position and pose, the second is the measurement vector. Since the measurements in this dissertation are restricted to those on a planar surface, the sensor position is represented by an x and a y coordinate relative to some frame of reference. The sensor pose is degrees relative to due north on the coordinate surface: -90° would be due west, while +/-180° is due south. If this vector were generalized to measurements in three dimensions, a z coordinate would be added to the position sub-vector, and an elevation scalar would be added to the pose sub-vector. The combined position and pose sub-vector will be represented as:

$$\langle x, y, \theta \rangle \tag{12.1}$$

Appropriately enough, x is also the sensor control vector. This vector contains a translation sub-vector and a rotational scalar. Once again, if extended to three-space, a fourth and fifth scalar would be added to represent location on the z axis and sensor elevation.

The measurement vector is more problematic. One of the things ardently to be wished for in a modular interface specification is universality: the same specification can

be used for every sensor.[48] This is clearly impossible, since some sensors report a scalar, others a vector. As described in Chapter 5, it is necessary to include an error estimate with each measurement and, in some cases, with each element in the measurement vector. For example, a voltage measuring sensor might report a single scalar representing the voltage at t_n, and a second scalar representing the measurement error in terms of statistical variance. A gray scale imaging sensor, on the other hand, returns an $n \times m$ vector of pixels. In the extreme case it might be necessary to return one vector of error terms for each pixel. A middle path between universality and a Tower of Babel of interface designs is to attempt to organize sensors into types sharing a single interface description – the fewer in total, the closer one comes to the ideal of universalizability.

It is easily argued that a certain set of sensors – ranging sensors – can be treated with a single, uniform, interface abstraction. Assuming, as above, that measurements are restricted to the x-y plane, there are only two measurements that we are interested in: target range r and target azimuth ϕ. Unlike sensor pose, which is relative to some absolute frame of reference, target azimuth is relative to sensor pose. The following equation shows how to convert target azimuth to absolute target azimuth.

$$\phi_{absolute} = \phi + \theta \tag{12.2}$$

---

[48] Universalizability is not merely an aesthetic goal, nor is simplicity a virtue in and of itself. As is shown in Chapter 5, the simpler the interface specification, the easier it is to achieve the second goal of this thesis: algorithm wrapping.

An error covariance matrix is associated with each measurement. Chapter 5 goes into some detail on the possible ways of representing the elements of this matrix. In the case of the ranging sensors used in these experiments, a simple 2×2 matrix **R** containing the co-variances of range and azimuth has proven sufficient. If the co-variance of the measurements is not uniform through the sensor space then the scalar can easily be replaced with a function of n variables. Thus when the information subscriber (or fusion algorithm) interrogates the virtual sensor for the measurement error, it returns a scalar result of the n variable function. For example, if the azimuth variance is some function of range, then the virtual sensor plugs the target range into the variance function and returns the resulting scalar value to the interrogator. The precision matrix ξ is implemented in similar fashion to **R**: $\mathbf{tr}(\xi) = \left\{\xi_r^2, \xi_o^2\right\}$. The off-diagonal elements are zero at layer n, and possibly non-zero at other layers. Since precision, like error, may be a function of some other parameter such as range, the virtual sensor interface may return the result of a function in ξ rather than constants. This is, in fact, the case with the laser range finder, where range precision is a function of range. (See Appendix C.)

The complete data vector appears in equation (13.3). The convention used in this dissertation is that each element of the vector is indexed by a unique sensor identifier s_n only when described individually. When treated as a vector, the vector variable itself will be indexed with the sensor identifier and a time index: s and k respectively.

$$\mathbf{S}_{s,k} = \langle x, y, \theta, r, \phi, \mathbf{R}, \xi \rangle \quad (12.3)$$

Appendix E: Dempster-Shafer Evidential Reasoning[49]

One criticism of probability based theories of evidence is that there is no simple way to distinguish between propositions for which there is no evidence (pro or con), and those for which there is equal evidence both pro and con. This can be seen quite vividly in the "non-informative" prior assigned cells in the section on Bayesian algorithms. The second item with which critics take issue is the property $P(c) = 1 - P(\sim c)$ as applied to belief functions or evidentiary functions.

One alternative to the probability based theories of evidence is the Demster-Shafer (D-S) model. D-S begins with a **frame of discernment** (FOD) Ω, which is an exhaustive, mutually exclusive set of propositions such as $\{a_1, a_2, a_3, \ldots a_n\}$. If A, B $\in 2^{\Omega}$ and A = $\{a_1, a_2\}$ and B = $\{a_3, a_4\}$, then ~A = $\{a_3, a_4, \ldots a_n\}$, A $\vee$ B = $\{a_1, a_2, a_3, a_4\}$, and so on. m: $2^{\Omega} \rightarrow [0, 1]$ is a mass assignment to the powerset of the FOD $\Lambda \subseteq 2^{\Omega}$ s.t. $m(\varnothing) = 0$ and $\sum_{A \subseteq \Lambda} m(A) = 1$. One example of a mass assignment might be: m_1(A, B, ~A) = <.3, .4, .2>. Note that $m_1(A) + m_1(\sim A) \neq 1.0$ as would be the case with probability assignments, since m_1 is really a belief assignment. Also note that m_1 assigns to the remaining portion

---

[49] The sources for this section were: [81], [51], [52], and [106].

of the FOD 1- $m_1(A) + m_1(\sim A) + m_1(B) = 0.1$. Finally, $m(B) = \sum_{A \subseteq B} m(A)$. In order to combine mass assignments, one employs the Dempster Rule:

$$m_1 \oplus m_2(A) = \frac{\sum_{B,C \in \Lambda : B \cap C = A} m_1(B) m_2(C)}{1 - \sum_{B,C \in \Lambda : B \cap C = \varnothing} m_1(B) m_2(C)}$$

Returning again to the range sensor example, the FOD would be: {E, F}, and Λ = (∅, {E}, {F}, {E, F}). Since $m(\varnothing)$ remains 0, and {E, F} = 0, we are concerned only with $m_i(\{E\})$ and $m_i(\{F\})$ where m_i is a mass assignment to the proposition "full" or "empty" for each cell i. Initially, m_i ({E}, {F}) = <0, 0>, representing complete ignorance of the state of affairs. We employ the same sensor model as used in the Bayesian example: $m_i(\{F\}) = 0.8$, $m_i(\{E\}) = 0.2$ if range = i; $m_i(\{F\}) = 0$, $m_i(\{E\}) = 0$ if range < i; and $m_i(\{F\}) = 0.2$, $m_i(\{E\}) = 0.8$ if range > i. For all cases, $m_i(\{E, F\}) = 0$. The first range reading is j.

1. For each cell $i > j$:

   $m_i(\{E\}) = ((0 \times 0) + (0 \times 0)) / (1 - (0.8 \times 0) + (0.2 \times 0)) = 0$,

   and $m_i(\{F\}) = ((0 \times 0) + (0 \times 0)) / (1 - (0.8 \times 0) + (0.2 \times 0)) = 0$.

2. For each cell $i < j$:

   $m_i(\{E\}) = ((0.8 \times 1) + (0 \times 0)) / (1 - (0.8 \times 0) + (0.2 \times 0)) = 0.8$,

   and $m_i(\{F\}) = ((0.2 \times 1) + (0 \times 0)) / (1 - (0.8 \times 0) + (0.2 \times 0)) = 0.2$.

3. For the single cell $i = j$:

$$m_i(\{E\}) = ((0.2 \times 1) + (0 \times 0)) / (1 - (0.8 \times 0) + (0.2 \times 0)) = 0.2,$$

$$\text{and } m_i(\{F\}) = ((0.8 \times 1) + (0 \times 0)) / (1 - (0.8 \times 0) + (0.2 \times 0)) = 0.8.$$

For the second range reading, assuming that it is also j, the update for the cell $i = j$ is:

$$m_i(E) = ((0.2 \times 0.2) + (0.2 \times 0) \ / (1 - (0.8 \times 0.2) \ + (0.2 \times 0.8)) = 0.0625,$$

$$\text{and } m_i(F) = ((0.8 \times 0) + (0.8 \times 0.8)) / (1 - (0.2 \times 0.8) + (0.8 \times 0.2)) = 0.94.$$

Appendix F: Algorithm Comparison

There are instances of each algorithm addressing the same problem domain and, in fact, one can see authors changing or swapping algorithms if one reviews their bibliographies from a historical perspective. Durrant-Whyte, for example, performs localization using Bayesian reasoning in [26], moves to Kalman Filters in a paper included in [1], and then on to Dempster-Shafer in [81]. Meanwhile Kurt Konolige published extensively using refinements of Elfes' Bayesian algorithms (e.g. [62]), but now seems to have moved on to Kalman Filters ([36]) and Markovian methods. The problem is, as pointed out in [22], there is a lack of empirical data comparing these algorithms.

> It was noticed that there were no papers comparing different methods on exactly the same problem and/or data, and that getting those might be illuminating. Since all have their own pet problem and software, and there may not be time to extend this, the idea arose to centralize the construction of such a testing facility.[50]

In fact, what little comparison there is appears contradictory. For example, in [81] they state: "... it is immediately apparent that the Dempster-Shafer approach provides [sic] much better solution to the map-building problem." In [51], however, they state:

---

[50] [22] p. 5.

> The remarkable thing about the two methods [D-S and Bayesian] is the similarity of the results they produce, with the Bayesian method appearing somewhat faster and slightly more efficient in its use of memory.[51,52]

One seminal paper analyzes four formalisms (D-S, Bayes, Fuzzy Set Theory, and Mycin) from a theoretical perspective, and concludes:

> The Bayesian calculus is well suited for applications where probabilities are known (or can be acquired with a reasonable effort). The calculus is especially attractive because of its strong theoretical foundation (note, however, that the calculus is generally unusable in its purest form due to exponential information complexity).
>
> The Dempster-Shafer calculus is a good choice for applications where uncertainty is best thought of as being distributed in sets rather than just single terms. Depending upon the particular domain, it can also have acceptable information and time complexity.[47]

In [38], a study comparing Bayesian probabilities, Dempster-Shafer belief functions, and MYCIN confidence factors resulted in the following comparison:

> In general, the MYCIN confidence factor methods stared well (i.e., with a higher level of accuracy) but leveled off after relatively few trials. Bayesian inference was relatively inaccurate at first, but improved rapidly and was best for large trial sets. Finally the Dempster-Shafer evidential process performed reasonably well under all trial set sizes.[38] p. 220.

In addition, there is a philosophical distinction between probabilistic formalisms and evidentiary formalisms. The latter are founded on the Dutch Book Theorem

---

[51] [51] p. 5.

[52] Since writing this, an unpublished draft experimentally comparing two fusion methods has come to our attention: [36].

> Let Σ be an algebra on Ω. The acceptance set $Acc_X(\Sigma, \Omega)$ is coherent *iff* bel_X is a probability function on Σ. [106]

This theorem supports the inference of $p(c)$ from $p(z)$, where z is evidence of the proposition c. It can be argued, and, as Voorbraak points out, has been argued, that a real valued function defined on $z \in Z$ is not interchangeable with a real valued function defined on $c \in C$.

Kalman Filters make the same demand as the Bayesian algorithms for known probabilities, but add to this a requirement for known covariances and a model of system and measurement noise. Also, while Bayesian Analysis, Fuzzy Set Theory, and Dempster-Shafer Evidential Reasoning have been used for sensor integration, Kalman Filters generally operate on commensurate data, and thus are not well suited for such applications as target recognition.

It should also be noted that the Bayesian and Dempster-Shafer methods contain an explicit sensor model based on probability of detection, while the Kalman Filter models the detection process as a measurement corrupted by noise.[53] This distinction can be made very clear (at a disservice to the Kalman Filter) by the following example. Suppose we set up a ranging sensor perpendicular to the flow of traffic on a divided highway,

---

[53] There are Bayesian formulations that incorporate a Gaussian noise term into the sensor model, but do not proceed to use this as a weighting factor for state estimate vs. measurement.

with one lane in each direction. The problem is to determine the distance from the sensor to each line of the highway. Assume also that cars are passing at fixed intervals: first one on the closer lane, then one on the farther lane. Thus at $k = \{0, 2, 4, 6, \ldots\}$ the cars pass on the lane closer to the sensor, while at $k = \{1, 3, 5, 7, \ldots\}$ the cars pass by on the far lane.

$$range(k) = \begin{cases} c_i & \text{if } k \text{ is even} \\ c_j & \text{if } k \text{ is odd} \end{cases}$$
$$\text{where } c_i < c_j$$

With the Dempster-Shafer and Bayes algorithms, and assuming a sensor model such as provided in the previous sections, as $k \rightarrow \infty$, the following probabilities obtain:

$$p_{\text{empty}}(c_m)_k = 1 \text{ where } m < i$$
$$p_{\text{occupied}}(c_m)_k = 0 \text{ where } m < i$$

These are the free space hypotheses, i.e., the probability that these cells are empty approaches certainty since all of our range data lies beyond them. Next:

$$p_{\text{empty}}(c_m)_k = p_{\text{empty}}(c_m)_0 \text{ where } m > j$$
$$p_{\text{full}}(c_m)_k = p_{\text{full}}(c_m)_0 \text{ where } m > j$$

Since all range readings are either at c_i or c_j, there is no evidence concerning cells at a greater distance, and thus these cell probabilities remain at their initial values. Next:

$$p_{empty}(c_m)_k = 0.5 \text{ where } m = i$$

$$p_{occupied}(c_m)_k = 0.5 \text{ where } m = i$$

This is because exactly half of the measurements show c_m as occupied, while the other half have it in the free space, and thus unoccupied. Finally:

$$p_{empty}(c_m)_k = 0 \text{ where } m = j$$

$$p_{occupied}(c_m)_k = 1 \text{ where } m = j$$

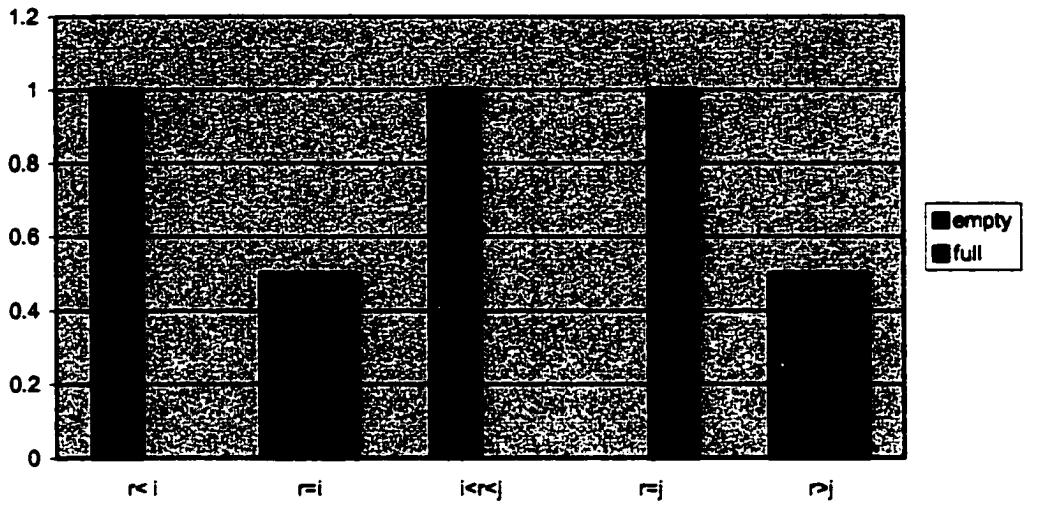

Figure 43. Occupancy Grid

Half of the measurements show c_j occupied, while the other half have c_j in the shadow of c_i, and thus there is no additional evidence either way. The resulting occupancy grid is given in Figure 43 using the Bayesian non-informative priors.

These results have a certain degree of intuitive appeal: whenever we can see cell c_j, it appears occupied. There is equal evidence that cell c_i is occupied and empty. If the final readings are thresholded appropriately, then the result is the range to both lanes.

Next, repeat the same experiment with a simple Discrete Kalman Filter. Since alternate measurements are of c_i and c_j, the optimum mean square estimate as $k \rightarrow \infty$ is $(c_i + c_j)/2$. Figure 44 shows the results where $c_i = 2$, $c_j = 4$, $R = 0.5$, and $k \rightarrow 30$. The range readings are corrupted with zero mean Gaussian noise with a variance of 0.5. The Gaussian noise was generated using the R250 Pseudo-Random Number Generator, ini-

tialized with the Park and Miller minimal standard generator [82], shaped by a Box-Muller transformation.[54] A total of 100 range readings were simulated, alternating between the near and far lanes. This sensor model resulted in only a single lane being estimated at the midpoint between c_i and c_j.

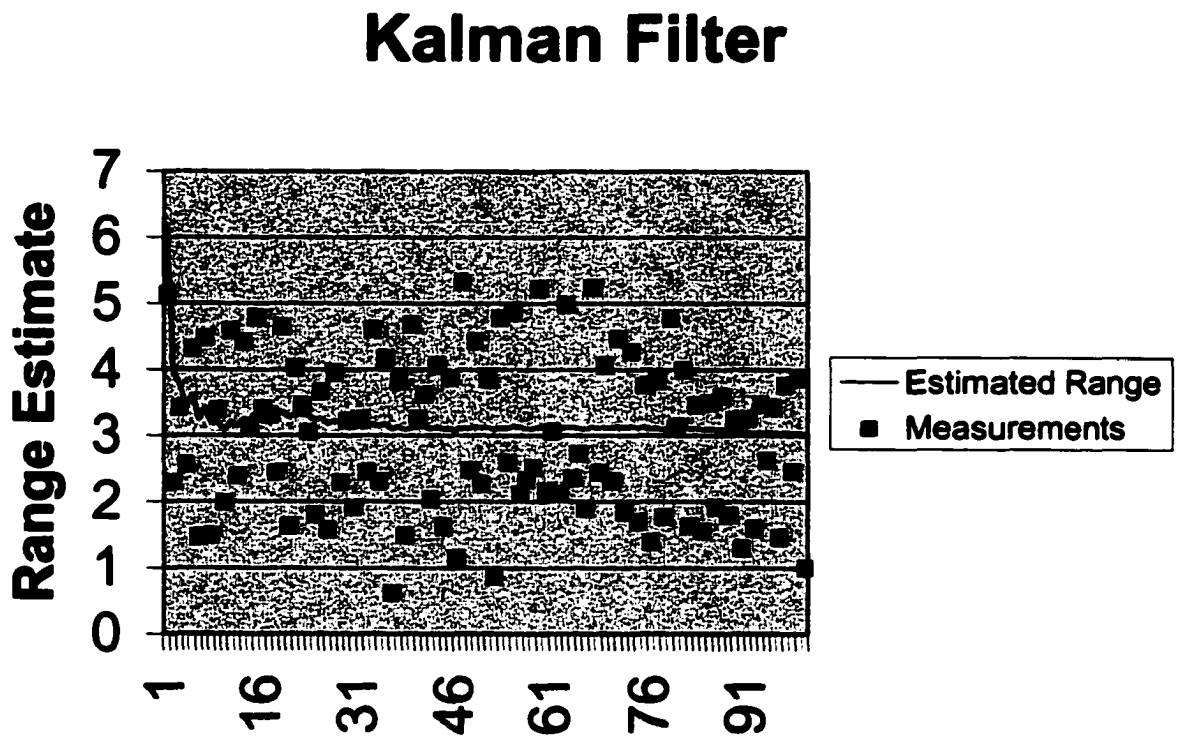

Figure 44: Linear Kalman Filter Range Estimates

This presents the primary challenge. Since there is no clear winner in the fusion algorithm arena, and there may never be one that is universally faster, more robust, and

[54] See http://www.taygata.com/random/gaussian.html.

accurate across the problem domain any generalizable sensor architecture must be designed to support all of the common fusion algorithms.

Appendix G. Test Tools

Nomenclature	Description	Purpose
Power Supply	Elenco Precision Quad Linear regulated (+5, +12, -12, 0-30 vdc)	Power supply for pan/tilt head, sonar, test equipment
Digital Volt Meter	Valhalla Scientific Model 4440 DVM with frequency counter	Verify power supply function, debugging
Digital Volt Meter	METEX ME-21 with PC Interface	Verify power supply function, debugging
Digital Logic Probe	Radio Shack 22-300	Debugging (sonar, parallel port interface)
Sonar	Milford Instruments Ultra-Sound Module 5-120	Sensor
Sonar	Polaroid 6500 Series Sonar Ranging Module	Sensor
CCD Camera	Connectix QuickCam	Laser Line-Striper
Laser	660nm, 4mW laser diode module, 30 degree line generator lens	Laser Line-Striper
Oscilloscope	Radio Shack ProbeScope	Debugging (sonar, parallel port interface)
Multi-Channel 10 bit Analog to Digital Converter (ADC)	BasicX-24	Sonar Controller
Parallel Port Debugger	Custom Built	Debugging interface logic

Table 10: Hardware Test Tools

Nomenclature	Description	Purpose
Debugger	DDD v.xxx	Debug code
Performance monitoring	xperfmon++ 1.1-2	System performance analysis
Performance monitoring	gprof	Code performance analysis
Performance monitoring	vmstat	System performance analysis
Bivariate Noise Generator	Mersenne Twister [74], Gnu Scientific Library Bivariate Distribution	Simulate correlated noise
MathCAD	MathCAD 2000	Rapid prototyping of simulation software

Table 11: Software Test Tools

Notational Conventions

General

Bolded lowercase Roman characters are vectors.
Bolded uppercase Roman and Greek characters are matrices.
Lower case Greek characters are angles.
Italicized characters are variables.

Specific

x_k	the value of x at step k
$\hat{\mathbf{x}}$	estimated state vector
θ	robot heading relative to some absolute frame of reference
$\mathbf{\Phi}$	state transition matrix
B	input transition matrix
G	process noise transmission matrix
H	measurement matrix
K_{xy}	covariance of k and y
P	state estimation error covariance matrix
r_k	the range measurement from sensor k
R	measurement noise covariance matrix
S_k	Sensor k
tr(A)	the trace of the matrix **A**
u	input vector
v	measurement noise vector
w	process or plant noise vector
x	x coordinate relative to some absolute frame of reference
x	state vector
X	a set of state vectors
y	y coordinate relative to some absolute frame of reference
ξ	precision
z	a measurement
z	measurement vector
Z	a set of measurement vectors
$\mathbf{z}_\mathbf{x}$	the measurement vector when the state vector is **x**

Ω	an arbitrary set
$p(x \mid y)$	the conditional probability of x given that y is the case

Bibliography

[1] Abidi, Mongi A. and Rafael C. Gonzalez. *Data Fusion in Robotics and Machine Intelligence.* Academic Press, Inc., Boston, MA, 1992.

[2] *American Institute of Physics Handbook.* 2nd Ed. McGraw-Hill, New York, 1963.

[3] Anderson, Brian D. O. and John B. Moore. *Optimal Filtering.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1979.

[4] Barker, Allen L., Donald E. Brown, and Worthy N. Martin. *Bayesian Estimation and the Kalman Filter.* IPC-TR-94-002. Institute for Parallel Computation, University of Virginia, 19 September 1994.

[5] Berchtold, Stefan, et al. Independence Diagrams: a Technique for Data Visualization. *Journal of Electronic Imaging*, Volume 9, Number 4, October 2000, Pp. 375-384.

[6] Borenstein, Johann and Yoram Koren. Histogramic In-Motion Mapping for Mobile Robot Obstacle Avoidance. *IEEE Journal of Robotics and Automation.* Volume 7, Number 4, 1991, Pp. 535-539.

[7] Borenstein, Johann and Yoram Koren. The Vector Field Histogram - Fast Obstacle Avoidance for Mobile Robots. *IEEE Transactions on Robotics and Automation.* Volume 7, Number 3, June 1991, Pp. 278-288.

[8] Borenstein, J., H. R. Everett, and L. Feng. *Where Am I? Sensors and Methods for Mobile Robot Positioning.* University of Michigan, April 1996.

[9] Boys, Richard J. Algorithm AS R80, A Remark on Algorithm AS 76: An Integral Useful in Calculating Noncentral *t* and Bivariate Normal Probabilities. *Applied Statistics.* Volume 38, Number 3, 1989, Pp. 580-582.

[10] Brown, Michael K. Feature Extraction Techniques for Recognizing Solid Objects with an Ultrasonic Range Sensor. *IEEE Journal of Robotics and Automation.* Volume RA-1, Number 4, December 1985, Pp. 191-206.

[11] Brown, C., et al. Distributed Data Fusion Using Kalman Filtering: A Robotics Application. *Data Fusion in Robotics and Machine Intelligence.* Academic Press, Boston, MA, 1992, Pp. 267-309.

[12] Chong, Kok Seng and Lindsay Kleeman. Indoor Exploration Using a Sonar Sensor Array: A Dual Representation Strategy. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1997.

[13] *A Culminating Advance in the Theory and Practice of Data Fusion, Filtering, and Decentralized Estimation.* The Covariance Intersection Working Group (CIWG), undated.

[14] Clarke, A. Bruce and Ralph L. Disney. *Probability and Random Processes, 2nd Edition.* John Wiley and Sons, Inc., New York, 1985.

[15] Crowley, James L. Mathematical Foundations of Navigation and Perception for an Autonomous Mobile Robot. *Reasoning with Uncertainty in Robotics.* Springer, Berlin, December 1995.

[16] Curry, David A. *Using C on the UNIX System.* O'Reilly &Associates, Inc., Sebastopol, CA,

[17] van Dam, Joris W. M., et al. *Neural Network Applications in Sensor Fusion for an Autonomous Robot.* Faculty of Mathematics, Computer Science, Physics, and Astronomy, University of Amsterdam, no date.

[18] Davies, E. R. *Machine Vision, 2nd Edition.* Academic Press, San Diego, CA, 1997.

[19] <deleted>

[20] Donald, Bruce and Jim Jennings. Sensor Interpretation and Task-Directed Planning Using Perceptual Equivalence Classes. *Proceedings of the 1991 IEEE International Conference on Robotics and Automation.* Sacramento, California, April 1991.

[21] Donald, Bruce and Jim Jennings. *Sensor Interpretation and Task-Directed Planning Using Perceptual Equivalence Classes.* Cornell University TR 91-1248, December 1991.

[22] Dorst, Leo, Michiel van Lambalgen, and Frans Voorbraak, ed., *Reasoning with Uncertainty in Robotics: International Workshop, RUR '95, Amsterdam, The Netherlands, December 1995, Proceedings*, Springer, Berlin, 1995.

[23] Dougherty, Edward R. and Charles R. Giardina. *Mathematical Methods for Artificial Intelligence and Autonomous Systems*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1988.

[24] Dudek, Gregory and Michael Jenkin. *Computational Principles of Mobile Robotics*. Cambridge University Press, Cambridge, UK, 2000.

[25] Durbin, James and John Shafer. *Connectix PC QuickCam Interface Specification*. version 1.3, Connectix Corporation, 24 September 1996.

[26] Durrant-Whyte, Hugh F.. *Integration, Coordination and Control of Multi-Sensor Robot Systems*. Kluwer Academic Publishers, Boston, MA, 1998.

[27] Ekstrom, Phil, Better Than Average. *Embedded Systems Programming*. Volume 13, Number 12, November 2000.

[28] Elfes, A.. Sonar-Based Real-World Mapping and Navigation. *IEEE Journal of Robotics and Automation*. Volume RA-3, Number 3, June 1987, Pp. 249-265.

[29] Elfes, A.. Multi-Source Spatial Data Fusion Using Bayesian Reasoning. *Data Fusion in Robotics and Machine Intelligence*. Academic Press, Inc., Boston, MA, 1992.

[30] Erkamp, Heleen. *A Representation in Sensor Values of an Unknown Environment through Exploration of a Mobile Robot*. Masters Thesis, Faculty of Mathematics and Computer Science, University of Amsterdam, August 1996.

[31] Fox, Dieter, Wolfram Burgard, and Sebastian Thrun. Markov Localization for Mobile Robots in Dynamic Environments. *Journal of Artificial Intelligence Research*, 11, 1999, Pp. 391-427.

[32] Gadre, Dhananjay V. *Programming the Parallel Port: Interfacing the PC for Data Acquisition and Process Control*. R&D Books, Lawrence, KS, 1998.

[33] Gardner, William A.. *Introduction to Random Processes with Applications to Signals and Systems*. Second Edition, McGraw-Hill, New York, 1990.

[34] Garvey, Thomas. A Survey of AI Approaches to the Integration of Information. *Multisensor Integration and Fusion for Intelligent Machines and Systems*, Ablex Publishing Corp., Norwood, NJ, 1995.

[35] Gat, Erann. Towards Principled Experimental Study of Autonomous Mobile Robots. *Autonomous Robots*. 2, Kluwer Academic Publishers, Boston, MA, 1995, Pp. 179-189.

[36] Gutmann, Jens-Steffen, Wolfram Burgard, Dieter Fox, and Kurt Konolige. An

Experimental Comparison of Localization Methods. Unpublished at this time.

[37] Hager, Gregory D.. *Task-Directed Sensor Fusion and Planning.* Kluwer Academic Publishers, Boston, MA, 1990.

[38] Hall, David L.. *Mathematical Techniques in Multisensor Data Fusion.* Artech House, Boston, MA, 1992.

[39] Hannebeck, Uwe D., Joachim Horne, and Gunther Schmidt. On Combining Set Theoretic and Bayesian Estimation. *Proceedings of the 1996 IEEE International Conference on Robotics and Automation.* April 1996, Pp. 3081-3086.

[40] Hannebeck, Uwe D. and Joachim Horne. A New Estimator for Mixed Stochastic and Set Theoretic Uncertainty Models Applied to Mobile Robot Localization. *Proceedings of the 1999 IEEE International Conference on Robotics and Automation.* May 1999, Pp. 1335-1340.

[41] Hannebeck, Uwe D. and Joachim Horne. Fusing Information Simultaneously Corrupted by Uncertainties with Known Bounds and Random Noise with Known Distribution. *Information Fusion* 1 (2000). Pp. 55-63.

[42] Haralick, Robert M. and Linda G. Shapiro, *Computer and Robot Vision, Volume 1*, Addison-Wesley Publishing Company, Reading, MA, 1992.

[43] Harmon, S.Y., G. L. Bianchini, and B. E. Pinz. Sensor Data Fusion Through a Distributed Blackboard. *Proceedings of the IEEE International Conference on Robotics and Automation.* 1986, Pp. 1449-1454.

[44] Hashemipour, Hamid R., et al. Decentralized Structures for Parallel Kalman Filtering. *IEEE Transactions on Automatic Control.* Volume 33, Number1, January 1988, Pp.88-94.

[45] Henderson, Thomas C. and Rod Grupen. Logical Behaviors. *Journal of Robotic Systems.* 7(3), 1990, Pp. 309-336.

[46] Henderson, Thomas C. and Esther Shilcrat. Logical Sensor Systems. *Multisensor Integration and Fusion for Intelligent Machines and Systems.* Ablex Publishing Corp., Norwood, NJ, 1995.

[47] Henkind, Steven J. and Malcom C. Harrison, An Analysis of Four Uncertainty Calculi. *IEEE Transactions on Systems, Man, and Cybernetics.* Volume 18, Number 5, September/October 1988, Pp. 700-714.

[48] Holt., B., J. Borenstein, Y. Koren, and D. Wehe. OmniNav: Obstacle Avoidance for Large, Non-circular Omnidirectional Mobile Robots. Presented at the ISRAM Conference, Montpellier, France, May 27-30, 1996.

[49] Hong, Lang. Distributed Filtering Using Set Models. *IEEE Transactions on Aerospace and Electronic Systems*. Volume 28, Number 4, October 1992, Pp. 1144-1153.

[50] Horton, J. W. *Fundamentals of Sonar*. United States Naval Institute, Annapolis, Maryland, 1959.

[51] Howard, Andrew and Les Kitchen. Generating Sonar Maps in Highly Specular Environments. *Proceedings of the Fourth International Conference on Control, Automation, Robotics and Vision*. December 1996, Pp. 1870-1874.

[52] Howard, Andrew and Les Kitchen. *Sonar Mapping for Mobile Robots*. Technical Report 96/34, Computer Vision and Machine Intelligence Laboratory, University of Melbourne, March 1997.

[53] Jetto, Leopoldo and Sauro Longhi. Development and Experimental Validation of an Adaptive Extended Kalman Filter for the Localization of Mobile Robots. *IEEE Transactions on Robotics and Automation*. Volume 15, Number 2, April 1999. Pp. 219-229.

[54] Jones, Joseph L. and Anita Flynn. *Mobile Robots*. A. K. Peters, Wellesley, MA, 1993.

[55] Julier, Simon J. and Jeffrey K. Uhlmann. A Non-divergent Estimation Algorithm in the Presence of Unknown Correlations. 1997.

[56] Kalman, R. E.. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*. Volume 82, March 1960, Pp. 35-45.

[57] Kamon, Ishay and Ehud Rivlin. Sensory-Based Motion Planning with Global Proofs. *IEEE Transactions on Robotics and Automation*. Volume 13, Number 6, December 1997, Pp. 814-822.

[58] Kanade, Takeo et al.. Development of a Video-Rate Stereo Machine. *1995 Conference on Intelligent Robots and Systems – IROS '95*, Volume 1, IEEE Computer Society Press.

[59] Kleeman, Lindsay and Roman Kuc. Mobile Robot Sonar for Target Localization and Classification. *The International Journal of Robotics Research*. Volume 14, Number 4, August 1995, Pp. 295-318.

[60] Klein, Lawrence A. *Sensor and Data Fusion Concepts and Applications*. SPIE Optical Engineering Press, Bellingham, WA, 1993.

[61] Koenig, Sven and Reid G. Simmons. Xavier: A Robot Navigation Architecture Based on Partially Observable Markov Decision Process Models. *Artificial In-*

telligence and Mobile Robots, AAAI Press/The MIT Press, Cambridge, MA, 1998, Pp. 91-122.

[62] Konolige, Kurt. Improved Occupancy Grids for Map Building. *Autonomous Robots* 4. 1997, Pp. 351-367

[63] Kuo, Benjamin C.. *Automatic Control Systems*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1987.

[64] L'Ecuyer, Pierre. Uniform Random Number Generators. *Proceedings of the 1998 Winter Simulation Conference*. 1998, Pp. 97-104.

[65] Lacroix, Simon and Gregory Dudek. On the Identification of Sonar Features. *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Genoble, France, 8-12 September 1997, Pp. 586–592.

[66] Le, Ha. Communication-Oriented Representation of Mathematical Objects. *Programming and Comput. Software (Programmirovanie)*, Volume 26, Number 1, January-February 2000.

[67] Lu, Feng and Evangelos Milios. Robot pose estimation in unknown environments by matching 2D range scans. *Proceedings CVPR '94., 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Pp. 935-938.

[68] Lu, Feng and Evangelos Milios. Globally Consistent Range Scan Alignment for Environment Mapping. *Autonomous Robots*. Volume 4, 1997, Pp. 333-349.

[69] Luo, R. C. and M. G. Kay. Data Fusion and Sensor Integration: State-of-the-Art 1990s. *Data Fusion in Robotics and Machine Intelligence*. Academic Press, Inc., Boston, MA, 1992.

[70] Luo, R. C. and Michael G. Kay. *Multisensor Integration and Fusion for Intelligent Machines and Systems*. Ablex Publishing Corp., Norwood, NJ, 1995.

[71] Lynn, Paul A. and Wolfgang Fuerst. *Introductory Digital Signal Processing with Computer Applications*. John Wiley and Sons, Inc., New York 1994.

[72] Marszalec, Janusz A. and Elzbieta A. Marszalec. *Integration of Lasers and Fiber Optics into Robotic Systems*. SPIE Optical Engineering Press, Bellingham, WA, 1994.

[73] Martin, Martin C. and Hans P. Moravek. *Robot Evidence Grids*. CMU-RI-TR-96-06, Carnegie-Mellon University, Pittsburgh, PA, March 1996.

[74] Matsumoto, Makoto and Takuji Nishimura. Mersenne Twister: A 623-

Dimensionally Equidistributed Uniform Pseudo-Random Number Generator. *ACM Transactions on Modeling and Computer Simulation*. Volume 8, Number 1, January 1998, Pp. 3-30.

[75] Maybeck, Peter S. *Stochastic Models, Estimation and Control*. Volume 1, Academic Press, Inc., Boston, MA, 1979.

[76] Moravec, H. P. and A. E. Elfes. High Resolution Maps from Wide Angle Sonar. *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, 1985, Pp. 116-121.

[77] Morgan, Dan. A Paean to Noise. *Embedded Systems Programming*. Volume 13, Number 2, Pp. 95-101.

[78] Murphy, Robin R. Dempster-Shafer Theory for Sensor Fusion in Autonomous Mobile Robots. *IEEE Transactions on Robotics and Automation*. Volume 14, Number 2, April 1998, Pp. 197-206.

[79] Musliner, David J. and Robert A. Larsen. The SMARTV Project: Smart Model Autonomous Real-Time Vehicles. Institute for Advanced Computer Studies, The University of Maryland, January 1995.

[80] Nakamura, Yoshihiko and Yingti Xu. Geometrical Fusion Method for Multisensor Robotic Systems. *Multisensor Integration and Fusion for Intelligent Machines and Systems*. Ablex Publishing Corp., Norwood, NJ, 1995, Pp. 241-259.

[81] Pagac, Daniel, Eduardo M. Nebot, and Hugh Durrant-Whyte. An Evidential Approach to Map-Building for Autonomous Vehicles. *IEEE Transactions on Robotics and Automation*. Volume 14, Number 4, August 1998, Pp. 623-629.

[82] Park, Stephen K. and Keith W. Miller, Random Number Generators: Good Ones Are Hard to Find, *Comm. ACM*. Volume 31, 1998, Pp. 1192-1201.

[83] Pearl, Judea. Bayesian and Belief-Functions Formalisms for Evidential Reasoning: A Conceptual Analysis. *Readings in Uncertain Reasoning*. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990, Pp. 540-574.

[84] Pearl, Judea, et al. Conditional Independence and Its Representations. *Readings in Uncertain Reasoning*. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990, Pp. 55-60.

[85] *Technical Specifications for the 600 Series Instrument Grade Electrostatic Transducer* and *Technical Specifications for the 6500 Series Sonar Ranging Module*. Polaroid Corporation, no date.

[86] *Polaroid 6500 Series Sonar Ranging Module PID 615077*. Polaroid Corporation,

no date.

[87] Press, William H. et al. *Numerical Recipes in C.* Cambridge University Press, Cambridge, GB, 1992.

[88] Quirin, William L. *Probability and Statistics.* Harper and Row, New York, 1978.

[89] Reid, I. D. and P. A. Beardsley. *Self-Alignment of a Binocular Robot.* Department of Engineering Science, University of Oxford, no date.

[90] Rodger, J. C. and R. A. Browse. An Object-Based Representation for Multisensory Robotic Perception. *Spatial Reasoning and Multi-Sensor Fusion: Proceedings of the 1987 Workshop.* Chicago. October 5-7, 1987, Pp. 13-20.

[91] Rosch, Winn L. *Hardware Bible.* Sams Publishing, Indianapolos, Indiana, 1997.

[92] *Report on Information Management to Support the Warrior.* United States Air Force Scientific Advisory Board, SAB-TR-98-02, 19 October 1998.

[93] Sabatini, Angelo M. A Statistical Method for Segmentation of Sonar Range Data. *Autonomous Robots.* Volume 1, 1995, Pp. 167-178.

[94] Saleem, S. K., et al. Electro-optical Sensor Fusion Tracker via a Set-Theoretic Approach. *Proceedings of the 33rd Conference on Decision and Control,* December 1994, Pp. 3794-3795.

[95] Saleem, S. K., et al. Minimum Volume Over-Bounding Ellipsoids for Set-Based Estimation in Target Tracking Applications. *Proceedings of the 35th Conference on Decision and Control.* Kobe Japan, December 1996, Pp. 2569-2570.

[96] Schollmeyer, Martina F. and William H.Tranter. Noise Generators for the Simulation of Digital Communication Systems. *Proceedings of the 24th Annual Symposium on Simulation.* 1991, Pp. 264-275.

[97] Schultz, Alan C. and William Adams. Continuous Localization Using Evidence Grids. Navy Center for Applied Research in Artificial Intelligence, no date.

[98] Schweppe, Fred C. *Uncertain Dynamic Systems.* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1973

[99] Shafer, Glenn. Belief Functions. *Readings in Uncertain Reasoning.* Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990, Pp. 473-481.

[100] *Sharp GP2D02 Compact, High Sensitive Distance Measuring Sensor.* Data sheet.

[101] Stevens, W. Richard. *UNIX Network Programming.* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1990

[102] Stevens, W. Richard. *Advanced Programming in the UNIX Environment.* Addison Wesley, Reading, MA, 1993

[103] Stewart, Zhahai. *IBM Parallel Port FAQ/Tutorial, Version 0.96.* 1 September 1994.

[104] van Dam, Joris W. M., et al. *Adaptive Sensor Models.* Faculty of Mathematics, Computer Science, Physics, and Astronomy, University of Amsterdam, no date.

[105] van Dam, Joris W. M., et al. *Neural Network Applications in Sensor Fusion for an Autonomous Robot.* Faculty of Mathematics, Computer Science, Physics, and Astronomy, University of Amsterdam, no date.

[106] Voorbraak, Frans, Reasoning with Uncertainty in AI. *Reasoning with Uncertainty in Robotics: International Workshop, RUR '95, Amsterdam, The Netherlands, December 1995, Proceedings.* Springer, Berlin, 1995, Pp. 52-90.

[107] Wallace, Andrew et al. Fusion of Intensity and Range Data. IEEE, 1994 (reprint).

[108] Welch, Greg and Gary Bishop. *An Introduction to the Kalman Filter, Department of Computer Science.* University of North Carolina at Chapel Hill, 1997.

[109] Yamauchi, Brian. *Frontier-Based Exploration: Laser-Limited Sonar.* http://www.aic.nrl.navy.mil/~yamauchi/frontier.

VITA

Thomas Franklin Litant

The author was born sometime during the last Millennium in New York City. He graduated from Lexington High School (Massachusetts), and then received a Bachelor of Arts in Philosophy and in English Literature (with distinction) from Colby College, Waterville Maine in 1976. He completed all requirements (except for dissertation) for a Ph.D. in Philosophy at Temple University. He chose to take a terminal Master of Arts in that field in order to relocate to Tokyo Japan in 1983. He received a Master of Science in Systems Management from the University of Southern California at Yokota Japan in 1987, and a Master of Science in Computer Science from the College of William and Mary in 1997. In 1997 he entered the Ph.D. program in the Computer Science Department at the College of William and Mary.

The author has been a Member of the Technical Staff of the MITRE Corporation since 1981. He is an Honor Graduate of the USAF Air Ground Operations School.